UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Arquitetura de Autoproteção para Internet das Coisas baseada no Laço MAPE-K

Dissertação de Mestrado

Ruan Marcos de Araujo Correia Mello

São Cristóvão – Sergipe

2017

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ruan Marcos de Araujo Correia Mello

Arquitetura de Autoproteção para Internet das Coisas baseada no Laço MAPE-K

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Admilson de Ribamar Lima Ribeiro
Coorientador(a): Edward David Moreno

São Cristóvão – Sergipe

2017

Ruan Marcos de Araujo Correia Mello
Arquitetura de Autoproteção para Internet das Coisas baseada no Laço MAPE-K/
Ruan Marcos de Araujo Correia Mello. – São Cristóvão – Sergipe, 2017-
87 p.
Orientador: Admilson de Ribamar Lima Ribeiro

Dissertação de Mestrado – UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, 2017.

1. Sistemas Autonômicos; 2. Auto-proteção; 3. IoT; 4. Laço MAPE-K. I. Orientador:
Admilson de Ribamar Lima Ribeiro. II. Universidade UFS. III. Título de Mestre

Ruan Marcos de Araujo Correia Mello

Arquitetura de Autoproteção para Internet das Coisas baseada no Laço MAPE-K

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Trabalho aprovado. São Cristóvão – Sergipe, 23 de Agosto de 2017:

Admilson de Ribamar Lima Ribeiro
Orientador

Edward David Moreno Ordonez
Coorientador

Ricardo José Paiva de Britto Salgueiro
Interno a instituição

Anderson Clay Nascimento
Externo a instituição

São Cristóvão – Sergipe
2017

Eu dedico essa tese a toda a minha família, amigos e professores que me deram o apoio necessário para chegar aqui.

Resumo

A Internet das Coisas é um novo paradigma baseado na Computação Ubíqua ou Computação Pervasiva. Computação Ubíqua e Computação Pervasiva que são termos utilizados para descrever a onipresença da informática no cotidiano das pessoas. O principal objetivo da Internet das Coisas é fazer com que as pessoas se comuniquem com as coisas e que as coisas também criem comunicação entre si sem necessidade da intervenção humana. O ambiente da Internet das Coisas possui bastantes restrições e a principal delas é o pouco recurso computacional dos dispositivos. O pouco recurso computacional dos dispositivos termina resultando em um ambiente muito inseguro e propício a diversos tipos de ataques, sejam eles físicos ou lógicos. Para tornar o ambiente da Internet das Coisas mais receptivo e bem visto por todos é importante investir em segurança. Para isso é muito interessante associar mecanismos de segurança com propriedades autonômicas, considerando o crescimento exponencial de dispositivos conectados. Este trabalho propõe uma arquitetura de segurança voltada ao ambiente da Internet das Coisas. A arquitetura proposta possui características autonômicas e é baseada no Laço de Controle MAPE-K. Para poder verificar a eficácia da arquitetura proposta foram abordados alguns dos principais ataques ocorridos no ambiente em questão (*Selective Forward, Blackhole, Sinkhole e Flooding*). Analisou-se o impacto causado por esses ataques e a interferência deles no funcionamento da rede criada a partir do protocolo de roteamento RPL.

Palavras-chave: Sistemas Autonômicos; Auto-proteção; IoT; Laço MAPE-K

Abstract

The Internet of Things is a new paradigm based on Ubiquitous Computing or Pervasive Computing. Ubiquitous Computing and Pervasive Computing are terms used to describe the omnipresence of information technology in people's daily lives. Its main goal is to create the possibility of communication between people and things and also between things without the need of human intervention. The Internet of Things environment has enough restrictions and the main one is the little computational resource of the devices. The little computational resource of the devices ends up resulting in a very insecure environment and conducive to various types of attacks, be they physical or logical. To make the Internet of Things environment more receptive and well-liked by all, it is important to invest in security. For this it is very interesting to associate security mechanisms with autonomic properties, considering the exponential growth of connected devices. This work proposes a security architecture focused on the Internet of Things environment. The proposed architecture has autonomic characteristics and is based on the MAPE-K Control Loop. In order to verify the effectiveness of the proposed architecture, it was approached some of the main attacks that occurred in the environment in question (Selective Forward, Blackhole, Sinkhole and Flooding). The impact of these attacks and their interference on the network operation created by the RPL routing protocol were analyzed.

Keywords: Autonomic Systems; Self-protecting; IoT; MAPE-K loop.

Lista de ilustrações

Lista de tabelas

Lista de códigos

Lista de abreviaturas e siglas

ACD	Algoritmo de Células Dendríticas
CAD	Channel Aware Detection
INTI	Intrusion detection of Sinkhole attacks on 6LoWPAN for Internet of Things)
CD	Células Dendríticas
CoAP	Constrained Application Protocol
IoT	Internet of Things
LLNs	Low Power and Lossy Networks
M2M	Machine to Machine (Máquina para Máquina)
MLPLW	Multi-layer Perceptron with Limited Weights
RFID	Identificação por Rádio Frequência
RPL	Routing Protocol Low Power
RSSF	Redes de Sensores sem Fio
SO	Sistema Operacional
IDS	Intrusion Detection System
6LowPAN	IPv6 Over Low Power Wireless Personal Area Networks
WSN	Wireless sensor networks
QoS	Quality of Service
LBR	LowPAN Border Router
MTU	Maximum Transmission Unit
IBM	International Business Machines
CPU	Central Processing Unit
DoS	Denial Of Service
RX	Taxa de recebimento
TX	Taxa de transmissão
ID	Identificação

Lista de símbolos

$\forall$	Para todo
$\sum$	Somatório
$\exists$	Existe
D	Valor deffuzyficado

Sumário

1

Introdução

A nova era da computação tende a estar além do tradicional ambiente de trabalho devido a integração recente que ocorreu entre sistemas embarcados, redes sem fio e Internet. Esta integração ocasionou o surgimento de um novo paradigma, a IoT (Internet das Coisas) que contempla um novo tipo de aplicação. O objetivo principal desse recente paradigma segundo os autores Gubbi et al. (2013) é fazer com que as pessoas se comuniquem com as coisas e que as coisas também criem comunicações entre si sem a necessidade de intervenção humana.

As aplicações IoT estão cada vez mais presentes e causarão grande impacto na vida das pessoas, porém apenas uma pequena parte delas está disponível para a nossa sociedade (ATZORI; IERA; MORABITO, 2010). Essas aplicações geralmente são executadas em um ambiente que contém restrição de recurso computacional. Apesar do avanço acelerado da IoT, os autores Atzori, Iera e Morabito (2010) dizem que existem grandes desafios a serem vencidos, por exemplo, a interoperabilidade dos dispositivos, o escasso recurso computacional e, sobretudo, a segurança.

De acordo com os autores Roman, Najera e Lopez (2011) a Internet e seus usuários já estão sob ataque constante, e uma crescente economia repleta de modelos de negócios baseia-se em prover o uso ético e seguro da Internet focando na exploração de fraquezas fundamentais da versão atual. Segundo Atzori, Iera e Morabito (2010) o ambiente da IoT é considerado ainda mais vulnerável que o ambiente da Internet convencional. Isso porque na maioria das vezes uma rede sem fio com um exagerado número de nós, facilita tanto o ataque físico quanto o ataque lógico já que não é possível implementar um esquema de segurança complexo devido a falta de recurso computacional. De fato, é previsto que surgirão modelos maliciosos e engenhosos voltados a esse ambiente, o grande desafio é impedir o crescimento de tais modelos ou pelo menos mitigar o seu impacto.

Dobson et al. (2010) destaca a importância das características autonômicas, levando em

consideração a crescente quantidade de dispositivos interligados no ambiente da IoT. Em 2011 a quantidade de dispositivos conectados já ultrapassava o real número de pessoas em todo o planeta e estima-se que em 2020 esse número deve chegar a 24 bilhões (GUBBI et al., 2013). Esses valores exorbitantes inviabilizam a gerência humana de segurança para os dispositivos, criando então a necessidade de automatizar essa função. De acordo com os autores Nami e Sharifi (2007) para resolver esse problema é essencial a utilização de uma propriedade que permita aos nós da rede, comunicação e reação aos ataques de acordo com as políticas de segurança definidas pelos usuários. Assim, os sistemas da IoT devem ser capazes de raciocinar de forma autônoma e tomar decisões de proteção.

O vice-presidente sênior da IBM (*International Business Machines*), Paul Horn, introduziu em março de 2001, pela primeira vez a utilização do termo Computação Autonômica. Horn escolheu deliberadamente um termo com uma conotação biológica buscando comparar em seu manifesto a necessidade de autogerenciamento em sistemas complexos que visam diminuir a carga dos administradores do sistema com a forma que o sistema nervoso autônomo regula a frequência cardíaca e a temperatura corporal. Dessa forma estaria libertando o cérebro consciente do fardo de lidar com estas e muitas outras funções que podem ser consideradas de baixo nível, mas de vital importância para funcionamento do mesmo (KEPHART; CHESS, 2003).

Esse manifesto apresentou as quatro propriedades da autogerência: autoconfiguração, auto-otimização, autocura e autoproteção. Também propôs no manifesto um modelo, o laço gerencial MAPE-K (*Monitor, Analyse, Plan, Execute and Knowledge*), onde divide as responsabilidades de cada elemento da computação autonômica (KEPHART; CHESS, 2003).

1.1 Problemática e Hipótese

Na IoT, tudo que é real torna-se virtual, isso significa que cada pessoa e coisa é localizável pois possui um endereço legível na Internet. Estas entidades virtuais podem produzir, consumir serviços e colaborar em direção a um objetivo comum (ROMAN; NAJERA; LOPEZ, 2011). A comunicação do mundo digital com o mundo real é realizada através de sensores e atuadores, por esse motivo, ela se torna ainda mais sensível do que a Internet convencional para ataques (XU; HE; LI, 2014). Além da possibilidade de acessar dados indevidos e esgotar energia ou memória dos dispositivos, é preciso se preocupar com o acesso aos dispositivos no mundo físico.

Xu, He e Li (2014), Gubbi et al. (2013), Roman, Najera e Lopez (2011) dizem que é necessário preocupar-se com a segurança e privacidade dos usuários. Essa preocupação impede a adoção massiva da IoT. Devido a isso é necessário prover um mecanismo de segurança para a IoT, com características autonômicas buscando reduzir a intervenção humana, que se tornará inviável com o crescimento exponencial de dispositivos conectados à mesma.

A partir do Algoritmo de Células Dendríticas (ACD), que terá a função de informar se a rede está sendo atacada ou não, é possível implementar um sistema autonômico com

características de autoproteção que forneça os mecanismos necessários de segurança para a IoT com um baixo consumo de energia e memória (ALMEIDA; RIBEIRO; MORENO, 2015). Em caso de ataque, após a detecção é possível descobrir qual tipo de ataque mais provável está ocorrendo na rede. Essa tarefa pode ser realizada através da Lógica Fuzzy que além de indicar qual o ataque mais provável irá informar o nó que possui mais chances de ser o atacante. Em seguida acredita-se que ao isolar o nó atacante podemos mitigar ou amenizar os danos causados à rede.

1.2 Objetivos

O objetivo deste trabalho é proporcionar uma arquitetura de segurança para a IoT e testar a sua eficácia em um ambiente simulado. No desenvolvimento dessa arquitetura foram aproveitados os módulos de Monitoramento e Análise do laço MAPE-K implementados pelos autores Almeida, Ribeiro e Moreno (2015) que utilizaram o Algoritmo de Células Dendríticas para detecção de ataques. Restaram os módulos de Planejamento e Execução para completar o laço MAPE-K, sendo assim esses módulos tornaram-se a essência do presente trabalho. Para alcançar o objetivo principal, foi necessário buscar alguns objetivos específicos. Os objetivos específicos desse trabalho são listados abaixo:

- Realizar levantamento das principais técnicas utilizadas para combater os ataques mais comum à rede da IoT (*SinkHole, Selective Forward, Hello Flood e Flooding*).:
- Realizar levantamento das ferramentas e materiais necessários para a simulação da arquitetura proposta.
- Implementar os ataques *Flooding, Sinkhole, Blackhole, Selective Forward* no ContikiOS para simulação no Cooja.
- Implementar a fase de Planejamento do laço MAPE-K na arquitetura proposta.
- Implementar a fase de Execução do laço MAPE-K na arquitetura proposta.
- Elaborar a análise da arquitetura desenvolvida quantitativamente – Memória consumida, Energia consumida, variação da taxa de serviço dos nós quando há ataque sem o sistema de proteção e quando há ataque com o sistema de proteção.
- Escrita da Dissertação apresentando resultados da análise da arquitetura.

1.3 Justificativa

Na Introdução deste trabalho foi visto que há inúmeros problemas quando se trata de IoT, desde a falta de recursos computacionais, interoperabilidade dos dispositivos, necessidade

de autogerenciamento ao se pensar na grande quantidade de nós conectados até a segurança e privacidade dos usuários. A Computação Autonômica se mostra um conceito importante para viabilizar a IoT, pois através dela será possível evitar a intervenção humana constante.

O laço MAPE-K proposto pela IBM (*International Business Machines*) busca dividir a carga de responsabilidades de cada elemento da computação autonômica entre quatro módulos mais uma área de conhecimento (Monitoramento, Análise, Planejamento, Execução, Conhecimento). Os autores Almeida, Ribeiro e Moreno (2015), construíram e analisaram os módulos de Monitoramento e Análise, abrindo caminho para trabalhos futuros onde os outros dois módulos devem ser implementados para fechar o laço MAPE-K.

O trabalho iniciado pelos autores Almeida, Ribeiro e Moreno (2015) foi simulado em duas plataformas uma plataforma tradicional e uma embarcada. A plataforma tradicional foi usada para a avaliação das redes neurais em relação ao seu desempenho, já a plataforma embarcada foi utilizada na análise da memória consumida. Para completar a arquitetura e testar sua eficácia de maneira completa será dada continuidade à implementação dos dois módulos que restam (Planejamento e Execução). Em seguida será criado outro ambiente simulado utilizando o simulador Cooja no ContikiOS. Ao finalizar este trabalho teremos resultados que poderão comprovar a necessidade de uma arquitetura de segurança específica para um ambiente com redes de baixa potência e perdas (LLNs).

1.3.1 Organização da Dissertação

Para facilitar a navegação e melhor entendimento, este documento está estruturado em seis capítulos, que são:

- Capítulo 1 - Introdução
- Capítulo 2 - Fundamentação Teórica
- Capítulo 3 - Trabalhos Correlatos
- Capítulo 4 - Arquitetura de Autoproteção para Internet das Coisas
- Capítulo 5 - Experimentos e Resultados
- Capítulo 6 - Conclusão

Na Introdução apresentamos o problema, a justificativa e o que foi proposto na dissertação. No capítulo Fundamentação Teórica são abordados os principais assuntos envolvidos no trabalho e tecnologias para contextualizar o leitor. No capítulo Trabalhos Correlatos são apresentados os trabalhos relacionados ao problema descrito, suas contribuições e limitações. No capítulo Arquitetura de Autoproteção para Internet das Coisas é apresentado de forma detalhada como irá funcionar cada módulo presente no Laço de controle. No capítulo Experimentos e Resultados é

descrito a metodologia do trabalho, o cenário das simulações e os dados utilizados para a coleta dos resultados.

2

Fundamentação Teórica

Neste capítulo há um tópico para cada um dos assuntos que viabilizaram a construção da nossa arquitetura de segurança, esses tópicos são abordados de maneira mais detalhada no decorrer do trabalho. Os tópicos presente neste capítulo correspondem respectivamente à: Internet das Coisas, Computação Autonômica, Laço de Controle MAPE-K, Sistemas Operacionais Embarcados, ConikiOS, Simulador Cooja, Protocolos de endereçamento IPv6 e 6LoWPAN, Protocolo de Roteamento RPL, Algoritmo de Células Dendríticas, Tipos de Ataques e Lógica Fuzzy.

2.1 Internet das Coisas

O termo Internet das Coisas cunhado por Kevin Ashton em 1999 (ASHTON, 2011) vem sendo muito comentado, por se tratar de um novo paradigma cujo conceito se baseia em fazer com que toda e qualquer coisa seja endereçada e ingresse a grande rede, a Internet. Os autores Roman, Najera e Lopez (2011) citam algumas tecnologias que servem como blocos de construção deste novo paradigma, como as RSSF (Redes de Sensores sem io), RFID (*Radio-Frequency IDentification*), serviços em nuvem e M2M (*Machine-to-Machine*). Além disso, este paradigma tem uma infinidade de domínios de aplicação como o automotivo, saúde, logística, monitoramento ambiental e muitos outros (ROMAN; NAJERA; LOPEZ, 2011).

Interligar à Internet uma variedade de coisas ou objetos do cotidiano para alcançar os objetivos desejados de maneira mais fácil e segura é sem dúvidas algo bastante tentador. Essa interligação é realizada através de sensores, atuadores, etiquetas RFID, dispositivos móveis, etc que interagem entre si, através de endereçamento único (XU; HE; LI, 2014).

A integração com a Internet implica que dispositivos irão utilizar um endereço de IP como um identificador único. Devido ao grande número de nós não é utilizado o IPv4. Segundo os

autores Wang, Zhong e Zhou (2012) todos os dispositivos e redes incorporadas são nativamente IP-ativado e conectados à Internet. O protocolo IPv6 portanto é considerado mais adequado do que o IPv4 para redes LoWPAN (*Low power Wireless Personal Area Networks*) por fornecer maiores espaços de endereçamento e melhores mecanismos de autoconfiguração. A aliança de trabalho 6LoWPAN (*IPv6 over Low power Wireless Personal Area Networks*) da IETF (*Internet Engineering Task Force*) é responsável por tornar possível a utilização do protocolo IPv6 nas redes IEEE 802.15.4 (WANG; ZHONG; ZHOU, 2012).

Essa interligação extrema vai trazer comodidade e economia sem precedentes, mas também vai exigir novas abordagens para garantir a sua utilização segura e ética (ROMAN; NAJERA; LOPEZ, 2011). Sensores, atuadores, etiquetas de RFID e as tecnologias de redes de sensores vão subir para enfrentar este novo desafio, em que os sistemas de informação e comunicação estão invisivelmente embutidos no ambiente que nos rodeia. Isto resulta na geração de grandes quantidades de dados que têm de ser armazenados, processados e apresentados de uma forma objetiva, eficiente e facilmente interpretável (GUBBI et al., 2013).

2.2 Computação Autonômica

A iniciativa da computação autônoma proposta pela IBM deriva em grande parte da sua inspiração baseada no sistema nervoso biológico (DOBSON et al., 2010). A ideia é que assim como os mecanismos do corpo que se auto ajustam não exigindo nenhuma ação consciente tais como o batimento cardíaco, atividade intestinal e secreção do hormônio sejam criados mecanismos que permitirão igualmente que um sistema de computador tenha uma autogestão (DOBSON et al., 2010).

De acordo com o manifesto proposto pela IBM em 2001 existem quatro propriedades autogerenciáveis: autoconfiguração, autocura, auto-otimização e autoproteção (KEPHART; CHESS, 2003).

- **Autoconfiguração:** Através da autoconfiguração um sistema autonômico irá se instalar e configurar-se para atingir as intenções do usuário de acordo políticas de alto nível pré-determinadas.
- **Autocura:** Através dessa propriedade é possível para sistemas autonômicos detectar, diagnosticar, e reparar problemas locais resultante de *bugs* ou falhas no *software* e *hardware*.
- **Auto-otimização:** A propriedade da auto-otimização está presente em sistemas que podem realizar alguma mudança no sistema de forma proativa para melhorar o desempenho e a qualidade de serviço.
- **Autoproteção:** A característica da autoproteção é atrelada a sistemas que se defendem de ataques maliciosos e de mudanças indevidas no sistema. O sistema autonômico com

autoproteção deve prevenir e antecipar falhas de segurança.

Dobson et al. (2010) afirma que os mecanismos de autogestão em uma computação autônoma não são entidades independentes. Por exemplo, o sucesso de um ataque ao sistema exigirá ações de autocura e uma mistura de autoconfiguração e auto-otimização, inicialmente para garantir confiabilidade e operação contínua do sistema e mais tarde para aumentar a autoproteção contra futuros ataques semelhantes.

2.3 Laço de Controle MAPE-K

O Laço gerenciável MAPE-K foi proposto pela IBM e tem por finalidade distribuir as tarefas de cada elemento da computação autonômica entre quatro módulos como pode ser visto na Figura 1, que são: monitoramento, análise, planejamento e execução (KEPHART; CHESS, 2003).

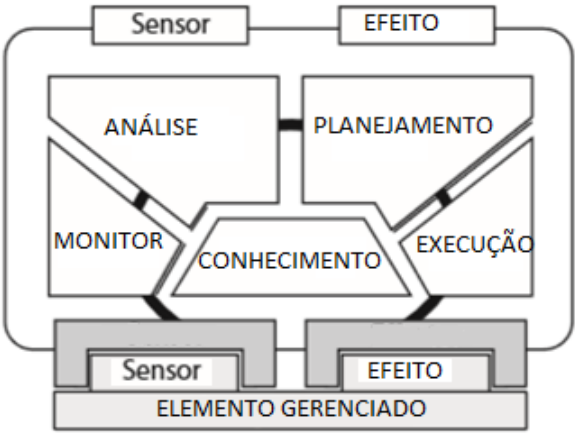
- **Módulo de Monitoramento:** Utiliza sensores para coletar dados a partir do elemento de gestão, o que poderia ser um recurso de *software*, *hardware*, ou um próprio gerenciador autônomo.
- **Módulo de Análise:** Fornece mecanismos para interpretar os dados recolhidos e prever as situações futuras.
- **Módulo de Planejamento:** Constrói as ações necessárias para atingir os objetivos específicos.
- **Módulo de Execução:** Utiliza ações predefinidas para realizar alterações em elementos gerenciados.

2.4 Sistemas Operacionais embarcados

Os Sistemas Operacionais (SO) voltados para *Desktop/Notebooks* como Windows, Linux, Mac OSX e os SOs IOS e Android que foram feitos para os poderosos dispositivos embarcados chamados *SmarthPhones* exigem bastante recurso computacional. Muito diferentes surgiram os SOs criados para os dispositivos chamados de *mote* que possuem muito pouco recurso computacional e uma grande necessidade de economia de energia.

Fica claro então que, o que torna os Sistemas Operacionais embarcados voltados para os *motes* tão diferentes é o *hardware* onde eles estão sendo executados. Os *motes* geralmente possuem um microcontrolador como CPU (*Central Processing Unit*), o que não é muito poderoso em termos de processamento, mas em termos de consumo de energia é muito bom pois consome muito pouca energia (REUSING, 2012). Entretanto, mesmo usando microcontroladores e outros

Figura 1 – Modelo de referência MAPE-K.



Fonte: Adaptado de (WEYNS; MALEK; ANDERSSON, 2010)

recursos de baixo custo de energia no *mote*, se for necessário o uso de todos eles em potência máxima, o consumo de energia passaria a ser relativamente alto, acabando assim com a bateria mais rapidamente (REUSING, 2012).

Quando o foco for a conservação de energia em um *mote*, o uso de um sistema operacional para melhor gerenciar os recursos e aperfeiçoar a economia da bateria é de extrema importância. Um dos SOs mais utilizados para o ambiente da IoT é o ContikiOS.

2.5 ContikiOS

Contiki é um sistema operacional projetado para ambientes com dispositivos que possuam restrições de memória e energia como os nós usados em WSN (*Wireless Sensor Networks*). O ambiente da IoT é composto por um grande número de minúsculos dispositivos de rede que se comunicam. Para redes de larga escala é importante a capacidade de baixar dinamicamente código para a rede (DUNKELS; GRONVALL; VOIGT, 2004). O Contiki se mostra eficiente para esse ambiente, pois, possui código fonte aberto, suporte para carregamento dinâmico, suporte para descarregamento de programas individuais e serviços, além disso, possibilita programação concorrente em cima de um *kernel* orientado a eventos (REUSING, 2012).

O Contiki é um sistema operacional implementado na linguagem C, inclui um simulador de rede chamado Cooja e foi portado para várias arquiteturas de microcontroladores, incluindo a *Texas Instruments MSP430* e o Atmel AVR (DUNKELS; GRONVALL; VOIGT, 2004). Outras informações relevantes sobre o sistema operacional retiradas do site do Contiki[1] são o uso dos padrões de Internet como o IPv6 e o IPv4 com eficiência energética e o suporte aos padrões de comunicação sem fio de baixo consumo de energia: 6LowPAN, RPL (Routing Protocol Low

[1] <http://www.contiki-os.org/>

Power) e CoAP (*Constrained Application Protocol*). Os dispositivos da IoT usualmente: (i) São dotados de menor capacidade de processamento, memória e energia; (ii) Realizam tarefas de forma colaborativa e (iii) executam sistemas eminentemente desenvolvidos usando a linguagem C ou sistemas operacionais baseados em C como o TinyOS, ContikiOS e Linux embarcado (DUNKELS; GRONVALL; VOIGT, 2004).

O *kernel* desse SO não oferece nenhuma abstração de *hardware*. É necessário implementar bibliotecas e *drivers* para realizar uma comunicação caso venha a ser necessário algum acesso ao *hardware*. O *kernel* do Contiki é orientado a eventos. Os processos só podem ser executados pelo *scheduler* quando se quer enviar um evento para o manipulador de eventos do processo ou quando é chamado pelo manipulador de *polling* (esse verifica periodicamente se há eventos vindos do *hardware*). Sendo assim os eventos não são preemptivos, ou seja, só haverá troca de processos pelo *scheduler* quando o processo que está usando a CPU for completado (REUSING, 2012). O Contiki não oferece funções de economia de energia explícitas. Ao invés disso, ele fornece aos dispositivos a possibilidade de colocar o microcontrolador ou os periféricos de *hardware* em modo ocioso. Logo, o controle de energia deve ser feito pelo programador, através da aplicação. Portanto, uma forma de programar a economia de energia seria gerenciando o tamanho da fila de eventos que o Contiki fornece, e caso ela esteja vazia, colocar a CPU em modo ocioso.

2.6 Simulador Cooja

Um simulador de rede pode ser usado para diferentes propósitos, por exemplo, o usuário pode apenas querer testar e depurar algum código de aplicativo, desenvolver aplicativos inteiramente novos ou avaliar algoritmos e protocolos de roteamento em ambientes simulados altamente personalizados. Pode também haver necessidade de adicionar novas funcionalidades, interações avançadas e análises de nós específicos em redes simuladas. O Cooja é um simulador voltado à IoT, seu principal objetivo é ser extensível e, ao mesmo tempo, utilizável (OSTERLIND, 2006).

O simulador Cooja se mostra bastante intuitivo por possuir um sistema de janelas e suporta os periféricos de *hardware* mais comuns, aumentando as chances de suas aplicações serem simuladas imediatamente sem muita intervenção. Os *plug-ins* incluídos no simulador Cooja básico permitem ao usuário interagir e visualizar informações durante e após a simulação. O usuário avançado pode adicionar novas interfaces personalizadas, estendendo os periféricos de *hardware* e interações de propriedade (apenas interfaces de simulação virtual), sem a necessidade de alterar nada no simulador de base. Ao ampliar *plug-ins* existentes, novas funcionalidades podem ser adicionadas rapidamente, ou ele pode criar *plug-ins* inteiramente novos, trabalhando com as ferramentas Java usuais para criar interfaces gráficas de usuário, a interface de simulação é implementada da mesma forma que um JPanel.

Considerando que as interfaces Cooja são a melhor maneira de interagir com nós simula-

dos, *plug-ins* são a melhor maneira para um usuário interagir com uma simulação. Os *plug-ins* são registrados em tempo de execução antes de serem usados, muitas vezes na inicialização do simulador (OSTERLIND, 2006). O usuário cria instâncias dos *plug-ins* registrados disponíveis durante as simulações. Alguns exemplos de *plug-in* do Cooja serão listados abaixo:

- **Cooja Timeline:** Cooja *Timeline* é um módulo que possibilita a visualização do tráfego da rede e do consumo energético dos dispositivos.
- **Log Listener:** *Log Listener* é responsável por capturar e exibir as mensagens de *log* iniciais.
- **Mote Interface Viewer:** Através do *Mote Interface Viewer* o usuário pode ver e interagir com interfaces de simulação.

2.7 Protocolos de endereçamento IPv6 e 6LoWPAN

O IPv6 é um protocolo de endereçamento criado para substituir gradativamente o IPv4 e possibilitar que a Internet continue crescendo por muito tempo. O IPv4 se mostra ultrapassado para redes LoWPAN basicamente pela quantidade de endereços livres disponíveis. O IPv6 traz um quadro de endereços livres disponíveis infinitamente maior e também possui melhores mecanismos de autoconfiguração (WANG; ZHONG; ZHOU, 2012).

O IPv6 expandiu o espaço de endereço IP de 32 para 128 bits e assim conseguiu aumentar bastante a quantidade de endereços livres. Reconhecendo o crescimento na largura de banda do *link*, foi necessário aumentar o requisito MTU (*Maximum Transmission Unit*) mínimo de 576 bytes para 1.280 bytes (MONTENEGRO et al., 2007). Para simplificar os roteadores e aumentar o desempenho, o IPv6 implementa a fragmentação nos nós de extremidade, em vez de nos roteadores intermediários. O componente IPv6, *Neighbor Discovery* (ND), usa o *multicast* de *link* local para resolução de endereços, detecção de endereços duplicados e descoberta de roteadores (NARTEN et al., 2007). A autoconfiguração de endereço sem estado (SAA) simplifica a configuração e o gerenciamento de dispositivos IPv6, permitindo que os nós se atribuam endereços significativos (CULLER; CHAKRABARTI, 2009).

Graças ao grande número de endereços disponíveis e características de autoconfiguração, o IPv6 se mostra o candidato ideal para redes sem fio de baixo consumo e baixa capacidade (LoWPANs). Porém para uma rede LoWPAN suportar o IPv6, alguns desafios devem ser vencidos devido as restrições de recursos computacionais e a natureza *multihop* (MONTENEGRO et al., 2007). O primeiro desafio a ser vencido é que pacotes IPv6 não estão naturalmente ajustados para LoWPANs. A baixa taxa de transferência, *buffer* limitado e quadros que são um décimo do tamanho do IPv6, exigência mínima MTU, tornam necessário a fragmentação e compressão de pacotes para realizar uma operação eficiente. O segundo desafio se dá pelo fato que o padrão 802.15.4 é de baixa potência, é mais propenso a interferências e falhas de link. Tais

características exigem que a camada de rede seja responsiva e adaptativa enquanto permanece eficiente em energia. Isso afeta todos os aspectos da rede, incluindo fragmentação, compressão, encaminhamento e roteamento.

Muitas aplicações embarcadas utilizam o padrão 802.15.4 que específica um link sem fio para LoWPANs. Essas aplicações se servem de uma grande quantidade de nós com escasso recurso computacional e que devem funcionar durante muito tempo utilizando modestas baterias sem sofrer interferências externas (CULLER; CHAKRABARTI, 2009). As muitas restrições já mencionadas anteriormente, levaram os fornecedores da LoWPAN a adotar protocolos próprios e soluções exclusivas de link (como o ZigBee), presumindo que o IP era muito volátil e requeria muita largura de banda (CULLER; CHAKRABARTI, 2009).

A extensão do IP para LoWPANs era considerada impraticável. Mas esforços do grupo IETF tornaram IPv6 viável para os links sem fio de comunicação de baixa potência, incluindo IEEE 802.15.4 (MONTENEGRO et al., 2007). O padrão proposto pelo IETF alterou drasticamente a visão dos fornecedores da LoWPAN, visto que, o objetivo do padrão é permitir a transmissão eficiente de pacotes IPv6 sobre links 802.15.4. Esse padrão foi chamado de 6LoWPAN (*Internet Protocol version 6 over Low power Wireless Personal Area Networks*) e fornece compressão de cabeçalho para reduzir a sobrecarga de transmissão e fragmentação para suportar o requisito MTU mínimo IPv6 (WANG; ZHONG; ZHOU, 2012).

2.8 Protocolo de roteamento RPL

Um fator muito importante relacionado ao consumo de energia para LLNs (Redes de Baixa Potência e Perda) é o meio de comunicação. Em redes constituídas por nós movidos a bateria, o ato de comunicar-se consome energia e os nós que se comunicam com mais frequência drenam sua fonte de energia mais rapidamente.

Para tornar a comunicação viável, primeiro é necessário estabelecer conexão e roteamento para envio e recebimento de mensagens. Uma das principais características das LLNs é a facilidade de perda de conexão, que termina inviabilizando a comunicação (WINTER, 2012). Em redes tradicionais, qualquer perda de conexão desencadeia o desejo de voltar a convergir rapidamente e encontrar caminhos de roteamento alternativos. Isso é necessário para fazer com que o tráfego de dados seja restabelecido o mais rápido possível resultando menos perdas em casos de falha na rede (VASSEUR et al., 2011). Porém disparar uma convergência completa com intuito de encontrar caminhos alternativos é bastante custoso em termos de recurso computacional. Além disso, o ato de procurar caminhos alternativos se tornaria muito constante, no caso de LLNs, causando falta de estabilidade na rede e esgotamento de recursos nos dispositivos envolvidos. O protocolo de roteamento voltado para esse ambiente deve estar preparado para lidar com essas características de rede.

O roteamento adequado para LLNs deve ser capaz de se autogerenciar e curar-se, sem

necessidade de intervenção manual. Esta necessidade se justifica pelo número elevado de dispositivos conectados nesse tipo de ambiente. A grande quantidade de dispositivos conectados torna impossível para um administrador de sistemas atribuir endereço IP manualmente ou digitar senhas que permitam acesso à rede. Além disso, é necessário que haja uma estratégia sofisticada de métrica de roteamento e restrições para a LLN. Métrica de roteamento é uma quantidade escalar usada como entrada para a melhor seleção de caminho (VASSEUR et al., 2011). Restrição, por outro lado, é usada como um critério adicional para podar links ou nós que não atendem ao conjunto de restrições (VASSEUR et al., 2011). Quanto ao endereçamento, o protocolo IPv6, se mostra o candidato ideal por ter grandes espaços de endereçamento e a capacidade de auto-configuração.

O IETF, grupo internacional que tem como principal missão, resolver problemas relacionados à internet e propor padronização de protocolos e tecnologias, reconheceu rapidamente a necessidade de formar um novo Grupo de Trabalho para padronizar uma solução de roteamento voltada a LLNs (WINTER, 2012). É importante ressaltar que essa solução de roteamento deve ser baseada em IPv6 para redes de objetos inteligentes. O grupo criado pela IETF foi chamado de ROLL (*Routing Over Low Power and Lossy*) em 2008 (WINTER, 2012).

O grupo de trabalho ROLL realizou uma análise detalhada dos requisitos de roteamento focando várias aplicações: redes urbanas, incluindo redes inteligentes, automação industrial, automação residencial e de edifícios. Este conjunto de aplicações é reconhecido como suficientemente amplo para cobrir a maioria das aplicações da IoT (VASSEUR et al., 2011). O objetivo era projetar um protocolo de roteamento para LLNs, que desse suporte as principais características da rede como: baixa largura de banda, perdas e baixa potência. O resultado do trabalho do ROLL foi o protocolo de roteamento RPL (*"Ripple"*), juntamente com especificações de suporte em métricas de roteamento, funções objetivas e segurança (WINTER, 2012).

Em suma o RPL é um protocolo de roteamento IPv6 para LLNs que especifica como construir um DODAG (Gráfico Acíclico Direcionado Orientado a Destino) usando uma função objetivo e um conjunto de métricas e ou restrições para calcular o melhor caminho. A função objetivo não especifica necessariamente a métrica e ou restrições, mas dita algumas regras para formar o DODAG (por exemplo, o número de pais, pais de *backup* e o uso de balanceamento de carga).

Segundo Vasseur (2011) o RPL difere de outros protocolos de roteamento que operam em ambientes menos restritos. Em LLNs, especialmente quando a rede é feita de dispositivos que devem economizar energia, é imperativo limitar o tráfego do plano de controle na rede. O gráfico construído pelo RPL é uma topologia de roteamento lógica construída sobre uma rede física para atender critérios específicos(VASSEUR et al., 2011). O administrador de rede pode decidir ter múltiplas topologias de roteamento (gráficos) ativas ao mesmo tempo, usadas para transportar tráfego com diferentes conjuntos de requisitos. Dessa forma um nó na rede pode participar, juntar um ou mais gráficos e marcar o tráfego de acordo com a característica do gráfico para suportar o

roteamento baseado em *Quality of Service* (QoS) e com restrições.

A partir desse parágrafo até o final da sessão é explicado, de acordo com o autor Winter (2012), como funciona a formação da topologia RPL e as regras que regem a construção do DODAG. O processo de construção do gráfico começa na raiz ou LBR (*LowPAN Border Router*), que é configurado pelo administrador do sistema. Lembrando que pode haver múltiplas raízes configuradas no sistema. O protocolo de roteamento RPL especifica um conjunto de novas mensagens de controle ICMPv6 para trocar informações relacionadas ao gráfico. Essas mensagens são chamadas DIS (*DODAG Information Solicitation*), DIO (*DODAG Information Object*) e DAO (*Destination Advertisement Object*).

Utilizando a mensagem DIO a raiz inicia o anúncio das informações sobre o gráfico. Os nós, vizinhos da raiz, receberão e processarão a mensagem DIO, potencialmente a partir de múltiplos nós, e tomarão uma decisão baseada em certas regras (de acordo com a função objetivo, as características do DAG, o custo do caminho anunciado e potencialmente a política local) para juntar-se ao DODAG ou não. Uma vez que o nó se juntou ao DODAG, ele terá uma rota em direção à raiz. O nó calcula o *rank* de si mesmo dentro do gráfico baseando-se no *rank* do nó que tomou como pai (inicialmente o nó raiz possui *rank* igual a zero e é o pai dos primeiros nós a ingressarem na rede). O *rank* do novo dispositivo no DODAG indica a "coordenada"do mesmo na hierarquia do gráfico. Caso o nó seja configurado para agir como um roteador, ele começa a anunciar as informações do gráfico com as novas informações para seus vizinhos. Se o nó é um "nó folha", ele simplesmente se junta ao gráfico e não envia nenhuma mensagem DIO. Os colegas vizinhos repetirão este processo e farão a seleção de pais, a adição de rotas e a publicidade de informações gráficas usando mensagens DIO. Esse efeito de ondulação cria as bordas do gráfico da raiz para os nós folha, onde o processo termina.

Na formação do DODAG, cada nó do gráfico tem uma entrada de roteamento em direção a seu pai (ou múltiplos pais dependendo da função objetivo) de uma forma *hop-by-hop* e os nós folha podem enviar um pacote de dados até a raiz do gráfico apenas encaminhando o pacote para seu pai imediato. Cada nó no gráfico possui um *rank* que representa a posição relativa do nó em relação à raiz na topologia do gráfico. A noção de *rank* é usada pelo protocolo RPL para vários propósitos, incluindo para evitar a criação de *loop*.

A mensagem DIS é usada pelos nós para solicitar proativamente informações gráficas (via DIO) dos nós vizinhos se eles se tornarem ativos em um ambiente de gráfico estável. Portanto, para um nó ingressar a rede, primeiro ele irá emitir mensagens DIS para os seus nós vizinhos e irá aguardar por uma mensagem DIO deles.

O tráfego que flui do nó folha em direção ao nó raiz é realizado de baixo para cima, mas há a necessidade do tráfego fluir também na direção oposta (de cima para baixo). Este tráfego pode originar-se de fora da LLN, na raiz ou em qualquer um dos nós intermediários e ser destinado a um nó folha. Para esse tráfego poder fluir de cima para baixo é necessário um estado de roteamento a ser construído em cada nó e um mecanismo para preencher essas rotas.

Essa tarefa é realizada pela mensagem DAO. As mensagens DAO são usadas para anunciar a acessibilidade de prefixo em relação aos nós folha possibilitando que o tráfego flua também para baixo. Essas mensagens carregam informações de prefixo, tempo de vida válido e outras informações sobre a distância do prefixo. À medida que cada nó se junta ao gráfico, ele enviará a mensagem DAO para seu pai com informações de todos os seus descendentes. A medida que cada nó recebe a mensagem DAO, processa as informações de prefixo e adiciona uma entrada de roteamento na tabela de roteamento. Esse processo continua até que as informações de prefixo cheguem à raiz e um caminho completo para o prefixo seja configurado. Este método é chamado de "armazenamento", pois cada um dos nós intermediários tem memória disponível para armazenar tabelas de roteamento.

Todo e qualquer nó que possua um pai na LLN envia uma mensagem DAO para o seu nó pai informando seus dados e os dados de seus filhos (caso exista), com intuito de permitir que o tráfego flua também para baixo. Essa forma de roteamento ficou conhecida como roteamento descendente. Isto significa que cada nó na rede teria de armazenar as informações de prefixo das mensagens DAO recebidas de cada um dos seus nós filhos. Isso causa implicações de memória e escalabilidade de tabela de roteamento em cada nó, uma vez que cada entrada de prefixo se traduz em uma entrada de roteamento na tabela de roteamento. Alguns nós na rede podem ter restrições significativas em relação à memória e podem ser incapazes de armazenar entradas de roteamento para rotas descendentes. Em virtude disso o protocolo RPL também teve que encontrar uma maneira de dar suporte aos nós que não possuem capacidade de armazenamento, criando um método chamado "não armazenamento" onde o nó intermediário não armazena nenhuma rota.

No modo de "não armazenamento", um nó usa mensagens DAO para informar seu DAO para seu pai seguindo assim até chegar ao nó raiz do gráfico. O nó raiz usa as informações recebidas para criar e gerir uma rota descendente. Os nós incluem as informações do seu nó pai no campo *"transit-info"* da mensagem DAO. Além disso, os nós podem empacotar DAOs enviando uma única mensagem DAO com várias informações de prefixo. Cada informação de prefixo pode ser associada com as suas próprias informações de trânsito. Neste modo de operação, espera-se que a raiz do DODAG tenha a capacidade de armazenar informação de encaminhamento enquanto os nós no DODAG operam no modo de "não armazenamento". Assim ao utilizar o modo de "não armazenamento", quando um nó qualquer da rede enviar um pacote para um outro nó dentro do domínio RPL, o pacote primeiro segue o gráfico até a raiz onde as informações de roteamento estão armazenadas. Neste ponto, a raiz do gráfico inspeciona o destino, consulta sua tabela de roteamento que contém o caminho para o destino graças às mensagens DAO que foram recebidas e roteia o pacote para seu destino usando um cabeçalho de roteamento específico para IPv6 (RH4) (HUI et al., 2010).

Um modo misto de operação não é permitido, ou seja, todos os nós do gráfico têm que operar apenas no modo de armazenamento ou não armazenando. Independente do modo utilizado

para gerar o gráfico o protocolo RPL não garante a ausência de *loops*, mas tenta evitá-los. É importante evitar os *loops*, pois eles causam perdas e congestionamento na rede. Uma das regras seguidas pelo RPL para evitar a formação de *loops*, depende da propriedade *"rank"* dos nós. Primeiramente não é permitido que um dispositivo selecione como pai um nó vizinho que tenha o *rank* maior que o seu. Em segundo lugar, não é permitido que um nó que deveria ter menor *rank* adquira um *rank* maior com intuito de aumentar o número de pais ou menor com intuito de aumentar o número de filhos.

Para evitar perdas e mau funcionamento da rede causado por falhas de link, indisponibilidade de dispositivos e má formação na árvore DODAG o RPL traz mecanismos de suporte que viabilizam o reparo no gráfico acíclico. O reparo é um recurso chave para qualquer protocolo de roteamento e tem como objetivo reconstruir parte ou toda a topologia de roteamento quando ocorrem falhas.

Na reconstrução de um DODAG o consumo de energia e memória dos dispositivos é bastante considerável em casos de LLNs. Portanto deve-se tomar cuidado e evitar o desencadeamento de uma reconstrução. O RPL especifica duas técnicas, conhecidas como reparo local e reparo global. Quando é detectada uma falha no link ou no nó e um dispositivo perde contado com o seu nó pai, um reparo local é acionado para encontrar rapidamente um caminho alternativo, escolhendo outro dispositivo como pai. Este é um reparo local sem implicação global em todo o gráfico. À medida que os reparos locais ocorrem, o gráfico pode começar a divergir de sua forma ótima. Caso seja necessário reconstruir todo o gráfico (DODAG) deverá ser utilizado o reparo global.

Reparação global é um mecanismo de reparo que reconstrói o gráfico a partir do zero. É uma técnica de otimização, mas tem um custo. O reparo global pode ser acionado somente a partir da raiz e tem um custo de tráfego de controle adicional na rede. Cada nó no gráfico irá executar novamente a função objetivo para seleção pai preferencial.

Depois de formado o DODAG, a comunicação P2P (ponto-a-ponto) torna-se possível, ou seja, qualquer nó pode se comunicar com qualquer outro nó no gráfico. Quando um nó envia um pacote para outro nó dentro da rede LLN, o pacote viaja "para cima" para um antepassado comum e em seguida ele é encaminhado na direção "para baixo"para o destino.

A segurança é crítica em LLNs, mas a complexidade para resolver esse problema é muito grande (devido a escasso recurso computacional), de tal forma que pode ser economicamente ou fisicamente impossível incluir sofisticadas provisões de segurança em uma implementação RPL. Além disso, muitas implementações podem utilizar a camada de ligação ou outros mecanismos de segurança para atender aos seus requisitos de segurança sem exigir o uso de segurança no RPL. Portanto, os recursos de segurança no RPL estão disponíveis como extensões opcionais.

2.9 Algoritmo de Células Dendríticas

O Algoritmo de Células Dendríticas (ACD) foi introduzido por Greensmith, Aickelin e Cayzer (2005) e é inspirado na Teoria do Perigo referente ao Sistema Imunológico Humano. O ACD é um algoritmo utilizado para realizar detecção de anomalias e alertar o sistema. Os principais elementos do ACD são as Células Dendríticas (CD), o Linfonodo e os antígenos. Os sinais de entrada da CD são: sinais de perigo, sinais Seguros, Sinais de PAMP, sinal de inflamação. Os sinais de saída da CD são: sinal de migração (*Costimulatory Molecules* – CSM), sinal semi-maduro e sinal maduro.

Iterativamente, os antígenos são apresentados às CD. Todos os sinais incrementam o sinal de migração, que indica quando a célula dendrítica irá migrar para o linfonodo e ser processada. Os sinais de perigo e PAMP incrementam o sinal maduro da CD enquanto o sinal seguro incrementa o sinal semi-maduro da CD. O sinal de inflamação potencializa o incremento de todos os sinais (GREENSMITH; AICKELIN; CAYZER, 2005).

Quando a CD atinge o limiar de migração será enviada para o linfonodo, e a CD será rotulada como madura, caso o sinal maduro seja maior que o semi-maduro, ou semi-madura, caso contrário. Quando o linfonodo possuir um determinado número de CDs, será calculado o índice de anomalia dos antígenos, o MCAV (*Mature Context Antigen Value*) a partir da fórmula:

$$MCAV = \frac{M}{(SM + M)} \tag{2.1}$$

Onde M é o número de CDs maduras e SM o número de CDs semi-maduras. Se o MCAV possuir um valor maior que o limiar determinado, o ACD detecta a presença de um invasor.

2.10 Tipos de Ataques

As características da rede IoT (baixa potência, energia limitada e poucos recursos) segundo Atzori, Iera e Morabito (2010) terminam expondo-a à várias ameaças. A maioria dos ataques no ambiente em questão, visam acabar com a energia limitada dos sensores, nesse caso, chamamos de ataque passivo, ou seja, um ataque que não modifica os dados. Já um ataque ativo, é aquele que modifica ou apaga dados trafegados na rede (ATZORI; IERA; MORABITO, 2010). Na sequência desta seção, são discutidos alguns dos ataques mais comuns em IoT e RSSF (MARTINS; GUYENNET, 2010).

- ***Selective Forward:*** Em um ataque *Selective Forward* o nó atacante realizando mau comportamento aceita o pacote de transmissão, mas recusa-se a transmitir determinados pacotes e simplesmente os descarta. O atacante deve escolher quais pacotes descartar de acordo com algum padrão, como tamanho, destino, ou origem (SHILA; CHENG; ANJALI, 2010).

- ***Blackhole:*** O ataque *Blackhole* é uma variação do *Selective Forward* que também é conhecido como *Grayhole*. No ataque *Blackhole* o nó atacante recebe os pacotes de transmissão e descarta todos os pacotes recebidos, independentemente do tipo, tamanho, origem ou destino (SHILA; CHENG; ANJALI, 2010).

- ***Sinkhole:*** Em um ataque *Sinkhole*, o atacante tenta atrair todo o tráfego dos nós vizinhos (GOYAL; BATRA; SINGH, 2010). Então, praticamente, o nó atacante escuta todos os dados transmitidos dos nós vizinhos. Somente este ataque não causa muito dano à rede, mas junto com outro tipo de ataque (*Selective Forward* ou *Blackhole*), pode se tornar muito poderoso.

- ***Flooding:*** Existem vulnerabilidades relacionadas à exaustão de memória e energia. Uma maneira de se aproveitar dessas vulnerabilidades é através do ataque *Flooding* que acontece quando um adversário envia muitas solicitações tentando estabelecer conexão com a vítima, cada solicitação faz com que a vítima tenha que alocar recursos na tentativa de manter a conexão (WOOD; STANKOVIC, 2002).

- ***Hello Flood:*** Um atacante *Hello Flood* usa um ou muitos nós maliciosos com um sinal poderoso, para enviar regularmente algumas mensagens na rede, com intenção de causar confusão na mesma (MARTINS; GUYENNET, 2010). Muitos protocolos de descoberta de rede *ad hoc* usam o envio de mensagem *Hello* para descobrir nós vizinhos e para criar automaticamente uma rede. Com o ataque *Hello Flood*, atacante pode usar um dispositivo com potência de transmissão grande o suficiente para comprometer cada nó da rede dentro de seu alcance. O atacante irá convencer os outros nós da rede que o invasor é o seu vizinho, para que eles tentem criar uma conexão, mas o invasor estará muito distante. Neste caso, o consumo de energia dos sensores é aumentando significativamente devido aos protocolos que dependem de informações de troca entre nós vizinhos para manutenção de topologia ou controle de fluxo (MARTINS; GUYENNET, 2010).

Anteriormente, vimos alguns dos ataques mais comuns em redes IoT em seguida são analisadas as possíveis estratégias para acabar ou mitigar os danos causados por eles.

Para atenuar os danos causados por ataques em uma rede, primeiro é necessário detectar esses ataques, utilizando um IDS (*Intrusion Detection System*). Um IDS monitora e analisa as atividades da rede, em seguida tenta detectar, com base nas informações colhidas, qualquer comportamento incomum que possa afetar a integridade da rede. A partir das informações fornecidas pelo IDS, são criadas estratégias para lidar com os ataques. Por exemplo:

- **Mitigando efeitos do *Selective Forward*:** Uma contramedida eficaz contra ataques *Selective Forward* é garantir que o atacante não possa distinguir os diferentes tipos de pacotes, forçando o atacante a enviar todos ou nenhum pacote (WALLGREN; RAZA; VOIGT, 2013). Nesse caso ao variar o ataque para o *Blackhole* essa contramedida não surtirá efeito.

- **Mitigando efeitos do *Sinkhole*:** Se as localizações geográficas dos nós do RPL DODAG são conhecidas, o efeito dos ataques *Sinkhole* pode ser atenuado pelo uso de controle de fluxo, certificando-se de que as mensagens estão viajando para o destino correto. O protocolo RPL também suporta várias instâncias DODAG oferecendo rotas alternativas para a raiz DODAG (HEER et al., 2011).

- **Mitigando efeitos do *Hello Flood*:** Uma solução simples para este ataque, é executar uma verificação bidirecional para cada mensagem "*HELLO*"(KARLOF; WAGNER, 2003). Se não houver reconhecimento, o caminho é assumido como ruim e um caminho diferente é escolhido. Se as localizações geográficas dos nós que compõem a DODAG RPL forem conhecidas, todos os pacotes recebidos a partir de um nó que está muito além da capacidade de transmissão de nó de rede comum, podem ser abandonados.

- **Mitigando efeitos do *Flooding*:** Limitar o número de conexões impede esgotamento de recursos por completo, que caso viesse a ocorrer iria interferir em todos os outros processos da vítima. No entanto, esta solução também impede que clientes legítimos criem conexão com a vítima causando problemas como filas (WOOD; STANKOVIC, 2002).

2.11 Lógica Fuzzy

Na computação, para se chegar à resolução de algum tipo de problema se envolve a manipulação de números e símbolos. Em contrapartida, os seres humanos empregam principalmente palavras que expressam valores escalonáveis de acordo com o raciocínio humano. Através de palavras expressas em linguagem natural partindo de percepções mentais é possível chegar à conclusão de um determinado problema. As palavras têm denotações difusas e podem ser categorizadas como representantes de valores lógicos dentro da lógica fuzzy (ZADEH, 1996). Nesta lógica, o raciocínio exato corresponde a um caso limite do raciocínio aproximado, sendo interpretado como um processo de composição de relações nebulosas (ZADEH, 1996).

Zadeh (1965) propôs uma forma mais generalizada de descobrir se um elemento pertence ou não a um determinado grupo, essa forma foi chamada de lógica fuzzy. O autor analisou que alguns elementos podem ter características que os aproximam ou distanciam de grupos específicos. De acordo com essas características os elementos podem assumir qualquer valor entre 0 e 1 como fator de pertinência, sendo que o valor 0 indica uma completa exclusão e um valor 1 representa completa pertinência. Esta generalização aumenta o poder de expressão da função característica.

A lógica nebulosa, que é outra forma de se referir à lógica fuzzy, segundo Zadeh (1996) já atingiu a maioridade e seus fundamentos se tornaram mais firmes, as aplicações cresceram em número e variedade, e sua influência nas ciências básicas especialmente nas ciências matemáticas e físicas tornou-se mais visível e mais substantiva. A utilização da lógica fuzzy em computadores

ou micro-controladores é muito bem aceita. Isso graças à simplicidade da implementação que pode vir a reduzir bastante a complexidade de um projeto (LEE, 1990).

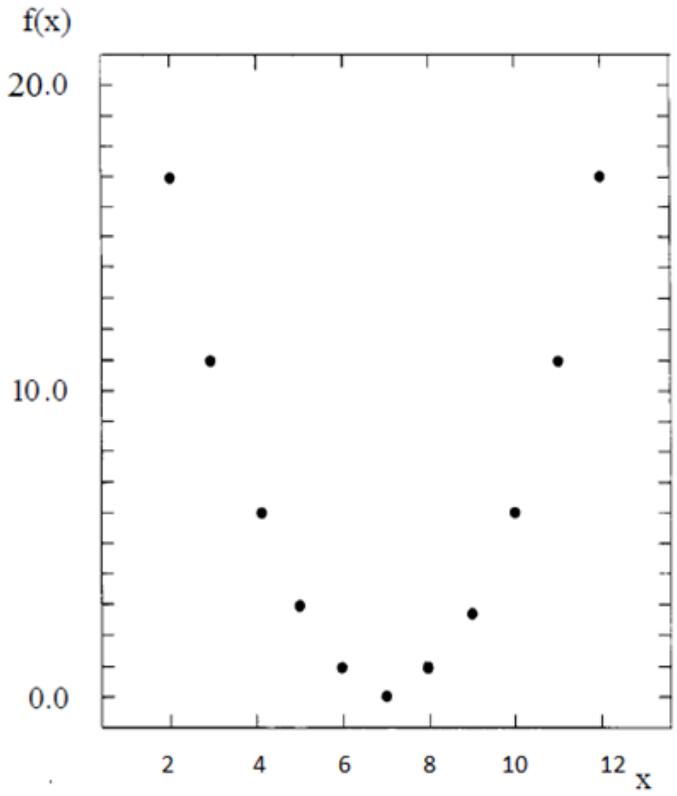
A lógica fuzzy difere dos sistemas lógicos tradicionais em características e detalhes (ZADEH, 1996). Nos sistemas lógicos multi-valores, o valor verdade de uma proposição pode ser ou um elemento de um conjunto finito, num intervalo, ou uma álgebra booleana. Por exemplo, em sistemas lógicos binários, o valor verdade só pode assumir dois valores (1 para sim e 0 para não), na lógica fuzzy o valor verdade de uma proposição pode ser um subconjunto fuzzy de qualquer conjunto predeterminado. Sendo assim de acordo com a classificação do subconjunto fuzzy é possível calcular o quanto o valor verdade se aproxima do valor "sim" e o quanto ele se distancia do valor "não" através de termos linguísticos. Na lógica nebulosa, os valores verdade são expressos linguisticamente (ex: muita verdade, verdade, pouca verdade, falso e muito falso), onde cada termo linguístico é interpretado como um subconjunto fuzzy do intervalo unitário (ZADEH, 1996). De acordo com o autor Zadeh (1996), existem muitas outras características da lógica fuzzy que se distinguem dos demais sistemas lógicos. A seguir são listadas algumas delas:

- Nos sistemas lógicos clássicos, o modificador mais utilizado é a negação enquanto que na lógica fuzzy uma variedade de modificadores de predicados é possível (ex: muito, mais ou menos, pouco). Estes modificadores são essenciais na geração de termos linguísticos (ex: muito alto, mais ou menos alto, perto, mais ou menos perto);
- Nos sistemas lógicos clássicos existem somente os quantificadores existenciais ($\exists$) e universais ($\forall$) já a lógica fuzzy admite, em adição, uma ampla variedade de quantificadores (ex: pouco, vários, usualmente, frequentemente);
- A probabilidade, no contexto da lógica clássica, é um valor numérico ou um intervalo. Na lógica nebulosa existe a opção adicional de se empregar probabilidades linguísticas (provável, altamente provável, improvável, etc), interpretadas como números fuzzy e manipuladas pela aritmética fuzzy.

É notório que na lógica fuzzy é muito mais comum utilizar palavras ou frases para definir um conceito, assumindo um sentido qualitativo, do que valores numéricos, que expressariam um sentido quantitativo (ZADEH, 1996). As palavras ou frases utilizadas para definir um conceito tornam-se variáveis linguísticas (ex: alto, médio, baixo, mais ou menos alto, mais ou menos baixo). Para atribuir um significado aos termos linguísticos, associa-se a cada uma das variáveis linguísticas um conjunto fuzzy definido sobre um universo de discurso comum. Para criar a associação entre as variáveis linguísticas e um conjunto fuzzy são criadas regras. Um exemplo de conjunto de regras aplicadas em sistemas clássicos da inteligência artificial pode ser visto na Figura 2.

A forma de definir um conjunto de regras vista na Figura 2 é eficiente e bastante simples, porém existem algumas desvantagens ao utilizá-la. Por exemplo, ao solicitar o valor de f(x)

Figura 2 – Conjunto de Regras aplicadas em sistemas clássicos da inteligência artificial.



Fonte: Adaptado de (GOMIDE; GUDWIN, 1994).

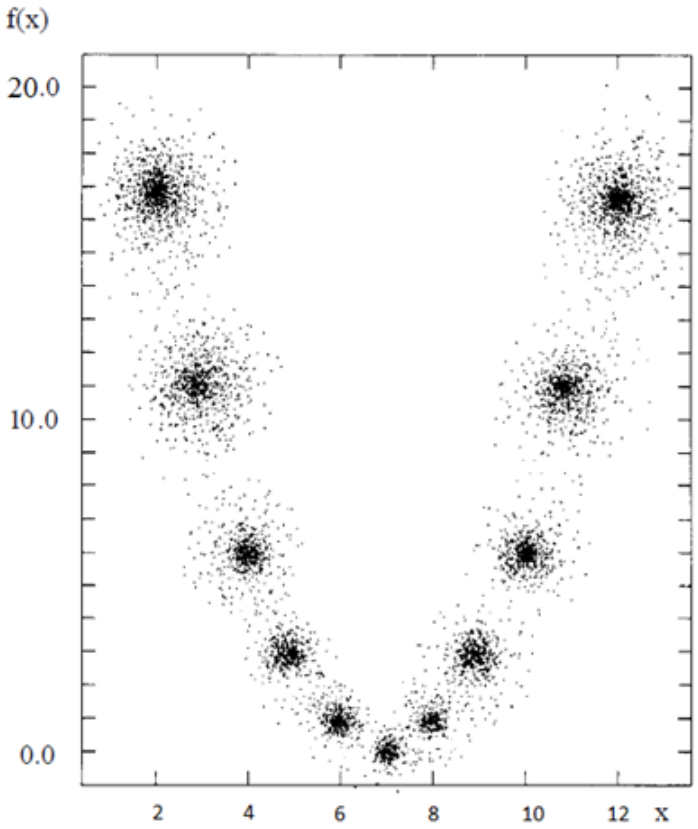
Tabela 1 – Conjunto de Regras

Regra 1	Se X = 2	Então f(x) = 17
Regra 2	Se X = 3	Então f(x) = 11
Regra 3	Se X = 4	Então f(x) = 6
Regra 4	Se X = 5	Então f(x) = 3
Regra 5	Se X = 6	Então f(x) = 1
Regra 6	Se X = 7	Então f(x) = 0
Regra 7	Se X = 8	Então f(x) = 1
Regra 8	Se X = 9	Então f(x) = 3
Regra 9	Se X = 10	Então f(x) = 6
Regra 10	Se X = 11	Então f(x) = 11
Regra 11	Se X = 12	Então f(x) = 17

quando x = 2, a conclusão que tiramos a partir das regras predefinidas que podem ser vistas na Tabela 1 será que f(x) = 11, mas no caso x = 2,46 a conclusão não pode ser encontrada, pois dentre as regras predefinas não existe um f(x) para x = 2,46. Portanto conclui-se que os sistemas clássicos binários não são muito eficientes com relação a dados imprecisos, com ruído ou com variação em valores de entrada.

Baseado na análise feita utilizando a Tabela 1 é possível afirmar que quando interpretamos o conjunto de regras de forma binária, foi necessário criar uma enorme quantidade de regras

Figura 3 – Conjunto de Regras interpretadas como regras fuzzy.



Fonte: Adaptado de (GOMIDE; GUDWIN, 1994).

Tabela 2 – Conjunto de Regras Linguísticas Fuzzy.

Regra 1	Se X está na vizinhança de 2	Então f(x) está na vizinhança de 17
Regra 2	Se X está na vizinhança de 3	Então f(x) está na vizinhança de 11
Regra 3	Se X está na vizinhança de 4	Então f(x) está na vizinhança de 6
Regra 4	Se X está na vizinhança de 5	Então f(x) está na vizinhança de 3
...		
Regra 12	Se X está na vizinhança de 12	Então f(x) está na vizinhança de 17

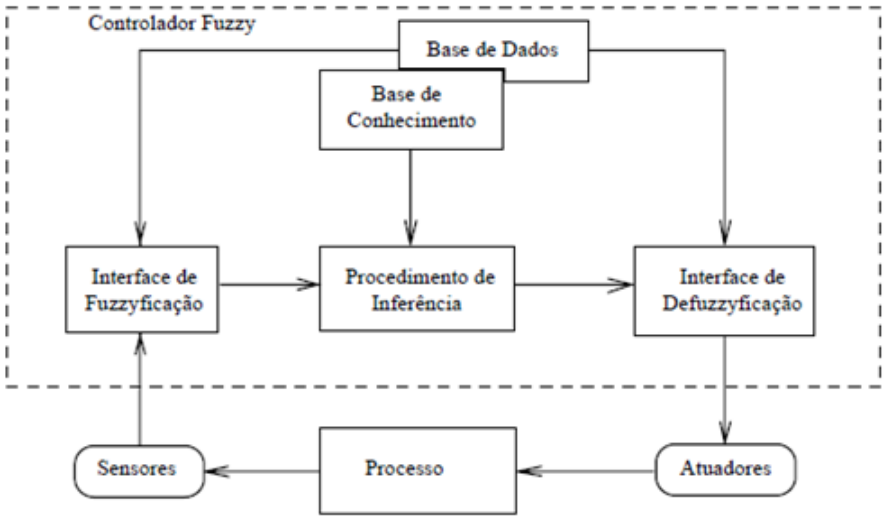
para se obter um resultado ou desempenho significativo. O mesmo conjunto de regras, porém interpretado como fuzzy, visto na Figura 3, é mais adequado ao se tratar de dados imprecisos, com ruído ou com variação nos valores de entrada. Isso porque a relação de regras deste conjunto fuzzy apresenta valores de saída razoáveis para todo e qualquer dado no universo de interesse como pode ser visto na Tabela 2 (ex: se x = 2,4 então f(x) está na vizinhança de 17). Isso é possível graças ao processo de fuzzyficação que transformou o valor exato visto na Figura 2 entre x e f(x) em um valor nebuloso (ZADEH, 1996). Sendo assim as regras irão indicar qual a variável linguística de saída após ocorrer a fuzzyficação e defuzzyficação.

Defuzzyficação é o processo de transformação dos valores nebulosos para uma das variáveis linguísticas predefinidas, ou seja, é o defuzzyficador que pesa as diversas respostas

fornecidas pelas regras lógicas e atribui à saída um número (ZADEH, 1996). Este número irá indicar o que é mais pertinente de fazer e com que grau (ex: “comprar” ou “não comprar”, “isolar” ou “não isolar”). Se o dado nebuloso estiver entre um determinado intervalo ele será representado por uma variável linguística caso contrario uma diferente variável linguística irá representá-lo.

Na Figura 4 é exibido todo o processo gerenciado por um controlador fuzzy, destacando os componentes principais: (i) A interface de fuzzyficação; (ii) A base de conhecimento; (iii) A base de dados; (iv) O procedimento de inferência e (v) a interface de defuzzyficação.

Figura 4 – Estrutura de Controlador Fuzzy



Fonte: Adaptado de (GOMIDE; GUDWIN, 1994).

A utilização da lógica fuzzy está muito difundida e tem gerado aplicações e produtos em diversas áreas como, por exemplo, no controle de processos industriais, na automatização residencial e nos reguladores de temperatura, umidade e volume água. GOMIDE, ROCHA e ALBERTOS (1992) afirmam que o potencial de manuseio de incertezas e de controle de sistemas complexos proporcionado pela lógica fuzzy vem sendo combinado com redes neurais artificiais as quais, por sua vez, possuem características de aprendizagem e adaptação. Esta simbiose vem gerando novas classes de sistemas e de controladores neurofuzzy, combinando assim os potenciais e as características individuais em sistemas adaptativos e inteligentes.

3

Trabalhos Correlatos

Neste capítulo serão abordados todos os trabalhos classificados como significantes e correlatos a presente dissertação. Foi realizada uma análise sistemática com intuito de encontrar trabalhos relevantes em bases consideradas importantes na área da computação (*IEEE Xplore*, *Science Direct* e a Biblioteca Digital Brasileira de Computação). Durante a execução da busca foram utilizadas as ferramentas de filtragem de cada base visando reduzir o número de trabalhos fora do escopo. Após realizar a busca foram aplicados alguns critérios para inclusão e critérios para exclusão de artigos. Os critérios de inclusão utilizados na análise sistemática foram: (i) Foram incluídos os artigos que tem por objetivo tornar o ambiente da IoT mais seguro; (ii) Foram incluídos os artigos que apresentaram técnicas utilizadas para detectar ou mitigar as possíveis ameaças e fraquezas da IoT. Os critérios de exclusão utilizados na análise sistemática foram: (i) Foram excluídos artigos que possuíam duplicidade; (ii) Foram excluídos artigos que não estavam acessíveis. Os trabalhos correlatos considerados mais significantes foram respectivamente SVELT de Raza, Wallgren e Voigt (2013), CAD de Shila, Cheng e Anjali (2010), Liu et al. (2011) e INTI de Cervantes et al. (2015).

3.1 SVELT

O nome dado ao IDS para a IoT projetado pelos autores Raza, Wallgren e Voigt (2013), foi SVELT. Este IDS é aplicado em redes 6LowPAN que utilizam o protocolo de roteamento RPL. A abordagem do SVELT é distribuída e centralizada, inserindo módulos no roteador de borda (6BR) e nos nós da rede. Os três principais módulos do SVELTE são: 6LoWPAN Mapper, Componente Detector de Intrusão e um *mini-firewall* distribuído. O 6LoWPAN Mapper (6Mapper) reconstrói a árvore de roteamento do protocolo RPL (DODAG) no 6BR, cada nó da rede possui um cliente para o 6Mapper. Para reconstruir a DODAG, é necessário enviar periodicamente requisições para os clientes 6Mapper, que responderão com suas informações

correspondentes: *NodeID*, *ParentID*, *Neighbor IDs* e *Rank* dos *Neighbors IDs*. Deve considerar se a rede usa alguma forma de autenticação de comunicação confiável, assim é possível reduzir o tamanho das requisições e respostas.

Uma vez que o SVELT detecta um ataque, o objetivo é mitigar seus efeitos e remover o intruso a partir da rede. Raza, Wallgren e Voigt (2013) dizem que a abordagem mais simples para remover um atacante é ignorá-lo. Tomar essa atitude requer a identificação do nó de ataque. Nem endereços MAC nem IP são confiáveis, já que eles podem ser facilmente falsificados. Os autores abordam duas opções para ignorar um nó, usar uma lista negra ou lista branca. A lista negra incluiria todos os nós maliciosos e uma lista branca incluiria todos os nós válidos. A manutenção de uma lista branca é mais fácil, na presença de muitos atacantes. Em virtude disso em SVELT é utilizado uma lista branca. As informações de localização dos nós também ajudarão SVELT para mitigar os sybil e *CloneID* ataques destinados a perturbar as informações de roteamento forjando identidades.

As implementações do SVELT e do *mini-firewall* foram feitas no ContikiOS, o código é aberto e disponível. Raza, Wallgren e Voigt (2013) utilizaram o ContikiOS e as suas implementações do RPL, do 6LoWPAN e da pilha IP. A avaliação foi feita de forma empírica, utilizando o Cooja, simulador de rede do ContikiOS, medindo a taxa de detecção e de verdadeiros positivos para cada experimento.

Os autores concluíram que o SVELT é muito efetivo para ataques *Sinkhole* em uma rede com poucas perdas, quando a rede possui mais perdas o sistema possui melhores resultados quando a rede RPL se torna estável. Conseguindo taxas próximas de 90% em uma rede com perdas quando a rede RPL se estabiliza e resultados próximos a 100% em uma rede sem perdas. Quanto ao consumo energético, a solução SVELT consome 30% mais energia do que utilizar apenas o RPL. Quanto ao consumo de memória, o cliente 6Mapper possui 1414 bytes de *overhead*, o cliente *Firewall* possui 246 bytes, o servidor 6Mapper varia com a quantidade de nós e vizinhos: 3580 bytes para 1 nó e 1 vizinho, 3846 bytes para 8 nós e 1 vizinho, 4152 bytes para 16 nós e 1 vizinho e 4724 bytes para 16 nós e 8 vizinhos.

3.2 CAD

Os autores Shila, Cheng e Anjali (2010) descrevem como desenvolveram uma forma de detectar, identificar o nó atacante e classificar o ataque em um ambiente de redes em malha sem fio (*Wireless Mesh Networks* - WMNs). Consideraram um caso especial de negação de serviço (DoS), conhecido como *Selective Forward* onde o nó atacante se comportando de maneira maliciosa encaminha apenas um subconjunto dos pacotes que recebe mas descarta os outros.

Enquanto a maioria dos estudos existentes sobre ataques do tipo *Selective Forward* se concentra na detecção do ataque partindo do pressuposto de um canal sem fio livre de erros, os autores Shila, Cheng e Anjali (2010) consideraram mais prático o cenário desafiador onde a

queda do pacote pode ser devido a um ataque, ou eventos normais de perdas, tais como colisão de acesso ao meio ou qualidade de canal ruim. Especificamente, desenvolveram um algoritmo que pode efetivamente diferenciar o *Selective Forward* das perdas de pacotes ocorridas em eventos normais de perdas. O algoritmo desenvolvido foi chamado de CAD (*Channel Aware Detection*) e é baseado em duas estratégias, estimativa de canal e monitoramento de tráfego. Se a taxa de perda monitorada em determinados saltos excede a taxa normal de perda estimada, os nós envolvidos serão identificados como atacantes. Além disso, realizaram estudos analíticos para determinar os limiares de detecção de ideais que minimizam a soma de alarme falso e probabilidades de detecção perdidas.

O procedimento de controle de trafego funciona basicamente da seguinte forma, cada nó intermediário ao longo de um caminho monitora o comportamento de seus vizinhos. Dado um caminho determinado pelo protocolo de roteamento, o CAD enviará mensagens ao longo do caminho para sondar, detectar possíveis ataques, em caso de uma detecção positiva, o CAD, então, aciona o protocolo de roteamento subjacente para ativar um novo processo de descoberta de rota.

No projeto CAD foi utilizado o *Network Simulator NS2 do Berkeley* (v2.29) para simulações. Ao final da simulação utilizando o algoritmo CAD, os resultados mostraram que tanto com a presença de eventos de perdas normais quanto sem, CAD pode detectar e classificar o ataque e descobrir os nós atacantes. Além disso, foi capaz de aumentar a taxa de entrega dos pacotes na rede para 97% de maneira eficiente em casos onde não haviam perdas relacionadas a eventos normais contendo apenas perdas relacionadas ao ataque que descartava 10% dos pacotes e para 92% quando descartados 50% dos pacotes. No caso de haver eventos normais relacionados a perdas de pacotes e o ataque *Selective Forward* simultaneamente, a taxa de entrega de pacotes com a utilização do CAD teve uma melhora de 69% para 78% quando houve 10% de pacotes descartados pelo ataque e uma melhora de 56% para 74% quando houve 20% dos pacotes descartados pelo ataque.

3.3 Liu et al.

Neste artigo, os autores Liu et al. (2011) propõem um método baseado em imunidade para a detecção de intrusão. Ao simular e definir os elementos imunes no ambiente da IoT, o método de aplicação baseado na teoria da imunidade é construído. Este trabalho apresenta características autonômicas, visto que, elementos de detecção no ambiente de detecção imune evoluem dinamicamente para simular mecanismos de auto-adaptação e auto-aprendizagem.

Os autores Liu et al. (2011) criaram um método matemático que é usado para deduzir o processo teórico da detecção de ataque. A análise teórica mostra que o método proposto fornece uma nova maneira efetiva para a tecnologia de detecção de intrusão no ambiente da IoT.

O método de detecção de intrusão construído para o ambiente da IoT foi validado através

de uma simulação onde um mecanismo é utilizado para reconhecer antígenos prejudiciais nos sistemas imunológicos. Os detectores simulam células imunes. Os antígenos simulam a assinatura de dados que são inseridos e produzidos a partir de dispositivos sensores. Os detectores são usados para classificar os antígenos em dados normais e dados de ataque para atingir o objetivo de detectar os ataques.

3.4 INTI

Neste trabalho os autores Cervantes et al. (2015) introduziram um sistema denominado INTI (*Intrusion detection of Sinkhole attacks on 6LoWPAN for Internet of Things*) para detectar ataques e em seguida isolar os atacantes da rede. A arquitetura utilizada nesse sistema foi preparada para atuar no ambiente da IoT utilizando os protocolos 6LoWPAN e RPL. O INTI estabelece o agrupamento dinâmico para suportar a transmissão de dados na IoT e observar o comportamento dos nós do roteador na tarefa de encaminhamento. Além disso, eles consideram o impacto da mobilidade do dispositivo, que é essencial em cenários urbanos, como cidades inteligentes. Além disso, o INTI visa mitigar os efeitos adversos encontrados no IDS que perturbam seu desempenho, como falso positivo e negativo, bem como o alto custo dos recursos.

O sistema combina estratégias de vigilância, reputação e confiança para a detecção de atacantes, analisando o comportamento de dispositivos. O ataque que o sistema em questão consegue detectar e mitigar os efeitos é o *Sinkhole*. Para isso, o nó que detectou o ataque *Sinkhole* gera e propaga uma mensagem de alarme em *broadcast* com o objetivo de alertar os nós vizinhos. Em seguida este nó promove o isolamento do atacante, enviando uma mensagem de restauração aos seus vizinhos.

O INTI é executado em quatro módulos: configuração de *cluster*, monitoramento de roteamento, detecção de ataque e isolamento de ataque. Os resultados mostram o desempenho e sua eficácia em termos de taxa de detecção de ataque, número de falsos positivos e falsos negativos.

Após realizar as simulações os resultados mostraram que o INTI garante uma taxa de detecção de pelo menos 90% com dispositivos fixos e 70% com dispositivos móveis na presença de nós maliciosos.

3.5 Considerações sobre os Trabalhos Correlatos

O SVELT, como os próprios autores afirmam, é o primeiro IDS para o ambiente da IoT. O trabalho apresentado possui uma enorme contribuição ao projetar um IDS com as características próprias de uma rede para IoT, considerando as tecnologias utilizadas na pilha de comunicação, como o 6LoWPAN e o protocolo de roteamento RPL. Porém a sua abordagem não busca a

autonomicidade da rede em se proteger de novos ataques, apenas pelo ataque determinado no nível de projeto da rede.

Assim como o SVELT o CAD não só detecta o ataque, mas também tenta mitigar os danos causados pelo atacante. O CAD é voltado para um ambiente de redes em malha sem fio (*Wireless Mesh Networks* - WMNs) e consegue diferenciar perdas ocorridas em eventos normais de um ataque legítimo do tipo *Selective Forward* de maneira eficiente.

O IDS projetado por Liu et al. (2011) se mostra muito interessante já que aborda características autonômicas e foi projetado para o ambiente da IoT. Além disso, foi inspirado na teoria do perigo do sistema imunológico humano e consegue se adaptar e aprender com o passar do tempo tornando-se cada vez mais eficaz. Porém os autores não especificaram quais ataques esse sistema foi capaz de detectar e nem apresentaram resultados vinculados a taxa de positivos ou falsos positivos relacionados à detecção.

O sistema INTI, projetado para o ambiente da IoT é utilizados não só para detecção do ataque *Sinkhole* como também para acabar com os danos causados pelo ataque na rede pelo. O INTI divide entre quatro módulos as tarefas necessárias para alcançar o seu objetivo. Essa divisão é importe e demonstra preocupação em relação a consumo de energia e memória dos dispositivos envolvidos.

Como pode ser visto na Tabela 3 a arquitetura que criamos demonstra ser mais completa em relação aos demais trabalhos. Nos capítulos seguintes serão descritos os detalhes da implementação da arquitetura de autoproteção e das simulações realizadas para extração dos resultados.

Tabela 3 – Comparação entre Trabalhos Correlatos

Qualidades	*SVELT*	*CAD*	*Liu et al.*	*INTI*	*Arquitetura Proposta*
Designado para IoT	X	-	X	X	X
Garante Autonomicidade	-	X	X	X	X
Detecta Sinkhole	X	X	Não Informou	X	X
Detecta Selective Foward	X	X	Não Informou	-	X
Detecta Blackhole	X	X	Não Informou	-	X
Detecta Flooding	-	-	Não Informou	-	X
Detecta Hello Flood	-	-	Não Informou	-	-
Mitiga efeito dos ataques	X	X	Não Informou	X	X

4

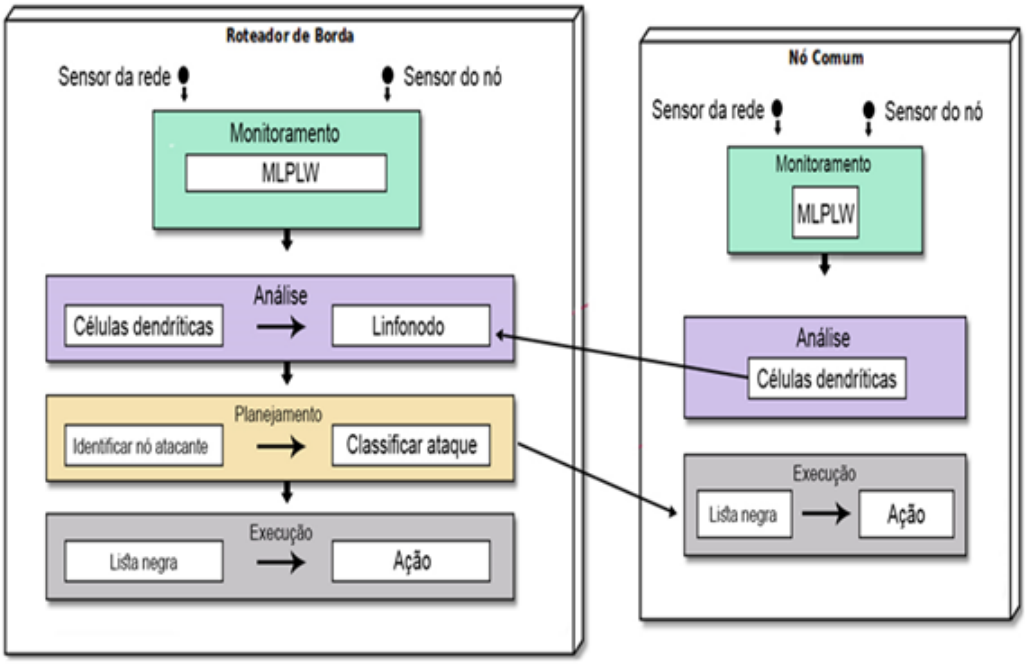
Arquitetura de Autoproteção para Internet das Coisas

A Arquitetura de Autoproteção para a IoT presente neste trabalho engloba o Laço de Controle MAPE-K proposto pela IBM em 2001 (KEPHART et al., 1999). O foco deste trabalho está direcionado aos módulos de Planejamento e Execução, visto que, os módulos de Monitoramento e Análise já haviam sido implementados pelos autores Almeida, Ribeiro e Moreno (2015). Porém, para melhor entendimento do funcionamento da arquitetura em questão, são detalhados todos os módulos presentes na mesma.

Em suma o módulo de monitoramento é responsável por coletar através de sensores que estão presentes nos nós da rede e no roteador de borda (6BR) dados referentes ao nó e à rede. No módulo de análise as informações capturadas são analisadas e indicaram a possibilidade de serem associadas a um ataque. Caso haja um ataque na rede os outros dois módulos (Planejamento e Execução) são acionados. As informações listadas à priori como dados relevantes para a análise são: (i) Tipo de protocolo de transporte; (ii) Número de mensagens enviadas; (iii) Número de mensagens efetivamente enviadas; (iv) Número de mensagens recebidas e (v) Número de mensagens perdidas.

Na Figura 5 é possível perceber que o nó de borda (6BR) possui todos os componentes. Isso ocorre pelo fato do 6BR possuir sensoriamento e também por ser o principal e mais robusto elemento da rede, por onde a maioria dos pacotes trafegados no protocolo RPL passa. Os nós da rede que tiverem o sistema de autoproteção possuem o componente de sensoriamento, monitoramento, planejamento, parte da análise, parte da execução e o componente de conhecimento. Os cinco módulos presentes na arquitetura são descritos de maneira mais clara a seguir.

Figura 5 – Arquitetura de Autoproteção para Internet das Coisas.



Fonte: Autoria própria

4.1 Módulo de Monitoramento

O componente responsável pela fase de monitoramento está presente onde houver sensoriamento, ou seja, nos nós da rede e no 6BR. Nesta fase, as informações da rede e dos nós são monitoradas. O principal componente da fase de monitoramento é a *Rede Neural Artificial Multi-layer Perceptron with Limited Weights* (MLPLW) que irá receber os dados do componente de sensoriamento e irá processá-los, fornecendo informações para o módulo de análise. Além disso, a MLPLW pode aprender padrões não-lineares durante a execução.

O MLPLW nesta arquitetura é usado principalmente para prever quais informações recebidas indicam um problema no nó ou na rede, por exemplo, o MLPLW pode prever a diminuição dos pacotes enviados com sucesso do endereço do remetente do pacote, se o treinamento for suficiente. A previsão do MLPLW é enviada para a fase de Análise, para o Algoritmo de Células Dendríticas. Os pesos da rede neural estão presentes no conhecimento referente ao monitoramento, porém o treinamento é feito no 6BR devido à limitação de recursos dos nós da rede.

4.2 Módulo de Análise

O componente responsável pela análise utiliza o Algoritmo de Células Dendríticas (ACD), que é inspirado no sistema imunológico humano, representado pela teoria do perigo. O ACD é usado para analisar os resultados recebidos do MLPLW e determinar se um ataque

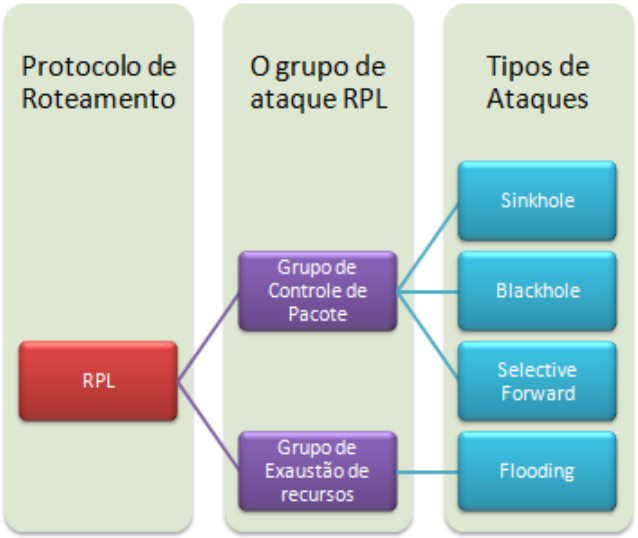
está acontecendo. As células dendríticas coletam informações do MLPLW e migram para a linfa quando a célula se torna madura ou semi-madura. A linfa recebe células dendríticas maduras e semi-maduras e determina se o sistema está em perigo.

Através desse algoritmo que apresenta uma alta distributividade e uma baixa taxa de falsos positivos, característica importante para um sistema de segurança, será possível deduzir se há ou não um ataque na rede.

4.3 Módulo de Planejamento

O componente responsável pelo planejamento necessita da confirmação de que há realmente um ataque na rede, caso contrário ele torna-se obsoleto já que seu principal objetivo é utilizar as informações adquiridas pelo módulo de análise para tentar descobrir quem é o atacante e qual o tipo de ataque sofrido pela rede. Tudo isso com intuito de decidir se deve ou não dar início a ação pré-definida no módulo de execução que virá a mitigar ou amenizar os danos causados pelo ataque. O módulo de conhecimento é responsável por manter todos os conhecimentos adquiridos pelo sistema. O conhecimento sobre o módulo de planejamento é mantido no 6BR. As informações mantidas pelo módulo de conhecimento são usadas para possibilitar a descoberta do tipo de ataque e do nó atacante. O tipo de ataque é classificado entre um dos dois grupos mencionados na Figura 6.

Figura 6 – Taxonomia dos Ataques RPL.



Fonte: Autoria própria

Através dos resultados relativos obtidos é possível apontar o grupo de ataque ao qual o ataque atual pertence. Por exemplo: (i) Em um ambiente em que esteja ocorrendo um ataque que pertença ao grupo de controle de pacote (*Sinkhole*), é levada em consideração a afluência de mensagens em nós legítimos e atacantes. Sendo assim é necessário avaliar a capacidade do

atacante influenciar a escolha da rota, comparando a afluência de mensagens em nós legítimos da rede em relação à dos nós atacantes, isto através de um número médio, mínimo, máximo e total de mensagens em nós atacantes e em nós legítimos; (ii) Caso seja um ataque que também pertença ao grupo controle de pacote, mas que contempla descarte de mensagens (*Blackhole* e *Selective Forward*) é levado em consideração número de mensagens enviadas, número de mensagens efetivamente enviadas e número de mensagens recebidas; (iii) Porém se o ataque pertencer ao grupo de exaustão de recursos (*Flooding*) é levado em consideração o número de pacotes enviados pelo nó, que tem intenção de enviar uma quantidade muito grande para sobrecarregar o sistema. Através desse processo será possível predizer também quem é o suspeito nó atacante.

Para criar uma forma de classificar a que grupo o ataque pertence observando os dados fornecidos, foi utilizado a lógica fuzzy. A lógica fuzzy funciona com intervalos de valores, e busca resolver os problemas de uma maneira que se assemelha a lógica humana (RODRÍGUEZ et al., 2014). A lógica fuzzy foi escolhida por poder lidar com problemas que envolvam dados imprecisos, vagos e incompletos. Uma proposição pode ser verdadeira, falsa ou ter um intermediário valor de verdade. Dessa forma a arquitetura é capaz de apontar o ataque e o atacante mais provável caso os dados não consigam levar a uma decisão precisa. Sistemas fuzzy tentam emular o processo cognitivo do cérebro com uma base de regras (RODRÍGUEZ et al., 2014). Portanto foi definida uma base de regras específica para resolver o nosso problema (identificar o nó atacante e o grupo de ataque).

Foram seguidas as seguintes etapas para completar o desenvolvimento da nossa lógica fuzzy.

- Identificação das variáveis de entrada e saída do processo;
- Definição das partições de cada variável linguística do sistema;
- Edição da Base de Regras Fuzzy;
- Compilação da Base de Regras;
- Edição das funções de pertinência associadas a cada termo linguístico previamente definido;
- Transferência da Base de Conhecimento do ambiente de desenvolvimento para o controlador Fuzzy;
- Execução e Depuração da Base de Conhecimento;
- Análise de Desempenho do sistema.

4.4 Módulo de Execução

O componente responsável pela execução terá como função mitigar ou amenizar os danos causados pelo ataque ocorrido na rede. Para que sejam realizadas ações com esse intuito, o módulo de planejamento precisa informar a que grupo o ataque que esta ocorrendo na rede pertence e quem é nó atacante. Essas informações são influenciadoras na ação predeterminada do módulo de execução.

O motivo pelo qual é necessário descobrir a identidade do atacante é para tentar ignorá-lo o mais rápido possível e assim evitar maiores danos à rede. Ao ignorar o nó malicioso é necessário que os nós afetados pelo ataque utilizem rotas alternativas, caso elas existam. A técnica escolhida para iniciar a mudança nas rotas foi o reparo global, que já é própria do protocolo RPL. O reparo global é uma técnica de otimização, mas tem um custo considerável em relação a recurso computacional dos dispositivos. Portanto só deve ser utilizada quando houver a certeza de que há um ataque ocorrendo na rede. Graças ao nosso IDS, composto pelos módulos de monitoramento e análise temos uma taxa de acerto na detecção de ataque muito próxima dos 100%, isso nos dá a confiança necessária para iniciarmos um reparo global quando preciso. Essa técnica reconstrói o DODAG RPL do zero sendo acionada somente a partir da raiz que irá propagar a informação de quem é o nó malicioso aos demais nós que forem ingressando à rede. Esses nós legítimos devem ignorar os DIS, DIO e DAO do nó malicioso identificado previamente. Sendo assim será evitado que o nó malicioso faça parte da rede.

A maneira como os nós da rede são identificados é muito importante, sejam eles legítimos ou maliciosos. Na nossa arquitetura a identificação foi feita através do endereço IP. Após identificar os nós da rede, três diferentes formas de isolar o nó malicioso foram analisadas antes de escolher a mais conveniente. As três formas analisadas foram:

- **Lista negra:** Após de identificar os nós e encontrar os atacantes, é criada uma lista e nela são adicionados todos os nós maliciosos. Dessa forma é possível excluí-los das possíveis rotas de tráfego de dados. Para ignorar o invasor, uma verificação é feita na Lista Negra, excluindo todos os nós encontrados do típico RPL DODAG que possui uma raiz e múltiplos nós.

- **Lista cinza:** Depois de identificar os nós e encontrar o invasor, é criada a lista cinza. O suspeito nó invasor é adicionado a esta lista com a intenção de excluí-lo das possíveis rotas de tráfego de dados, por um tempo predeterminado. Após o fim do tempo predeterminado, o suspeito nó invasor é excluído da lista cinza. Desta forma, se houver alguma dúvida sobre a identidade do nó atacante, o nó pode voltar a juntar-se à rede. Caso haja certeza que o nó em questão é o atacante, o mesmo é adicionado permanentemente à lista. Para ignorar o suspeito invasor, ao criar o roteamento, uma verificação é feita na Lista cinza, excluindo todos os nós encontrados do típico RPL DODAG que possui uma raiz e múltiplos nós.

- **Lista branca:** Como no exemplo da Lista negra, uma lista é criada depois de identificar os nós e encontrar o nó atacante. Mas, desta vez, é adicionado à Lista branca, somente os nós válidos e todos os nós maliciosos são excluídos. Desta forma, é necessário que haja uma verificação indicando quais nós são válidos e devem pertencer ao típico RPL DODAG com uma raiz e vários nós.

Em virtude de maior facilidade de manutenção, foi escolhida a Lista negra. Essa lista possui um gerenciamento mais simples, pois, geralmente existe uma quantidade pequena de nós adicionados a ela. Isso por que quase sempre a quantidade de nós válidos é superior a de nós maliciosos. Além disso, para armazenar todos os nós válidos no caso da Lista branca seria consumido mais espaço de armazenamento nos dispositivos.

4.5 Módulo de Conhecimento

O módulo de conhecimento presente no laço em questão tem por finalidade armazenar o conhecimento do sistema e está distribuído pela rede. O conhecimento relativo ao módulo de planejamento e de execução está presente no 6BR. As informações guardadas e definidas como conhecimento do sistema são utilizadas para facilitar e acelerar tanto a descoberta do tipo de ataque, quanto à ação a ser tomada para proteger a rede.

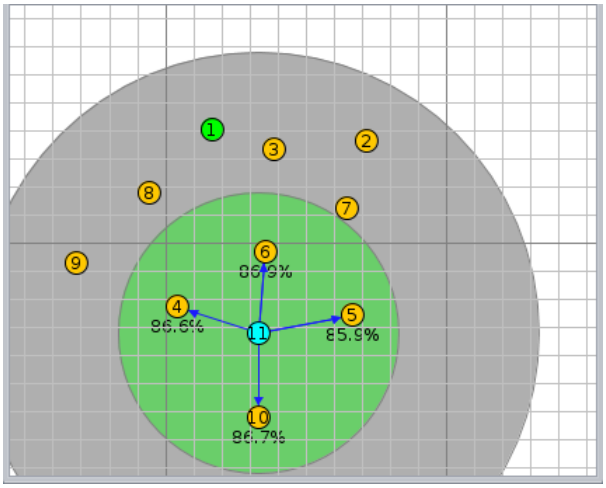
5

Experimentos e Resultados

Para validar a nossa arquitetura foram realizadas algumas simulações para coletar os dados de um cenário criado sem a presença de ataque. A intenção é comparar os resultados obtidos com os resultados de outros cenários também simulados sem utilizar o sistema de autoproteção e utilizando o sistema de autoproteção na presença de ataques.

As simulações foram realizadas utilizando o Cooja, simulador do ContikiOS. Foram implementados e simulados quatro diferentes ataques *Flooding, Sinkhole, Selective Forward e Blackhole*. As simulações com e sem o nó atacante foram formadas seguindo uma topologia DODAG em que há um certo número de nós, sendo um deles a raiz. A simulação da rede criada no Cooja pode ser vista na Figura 7.

Figura 7 – Rede simulada no Cooja.



Fonte: Autoria própria

Definimos o número de nós presentes na simulação e todos usaram a mesma plataforma (*Skymote*). O protocolo de roteamento utilizado foi o RPL e para o endereçamento

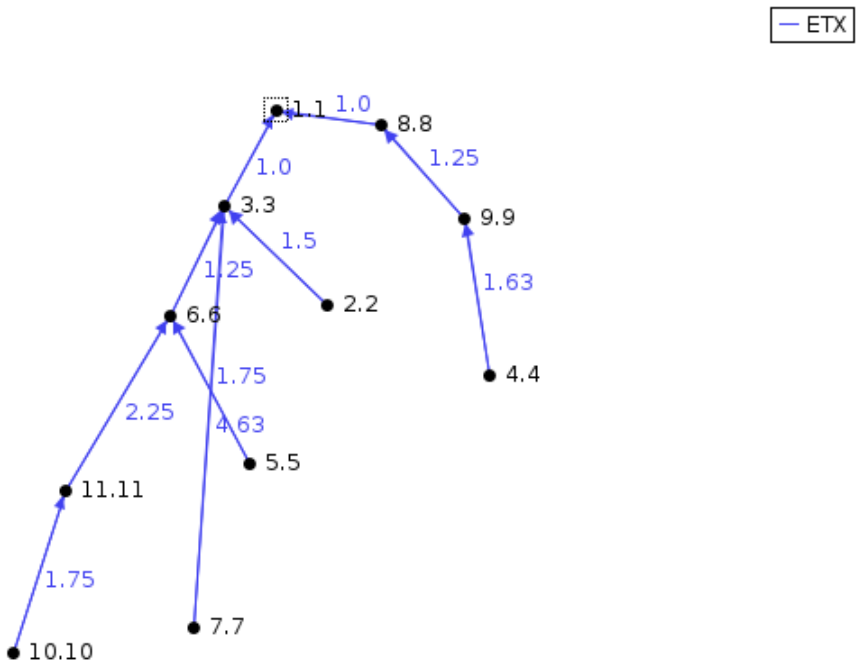
foi utilizado 6LoWPAN que garante a utilização do IPV6 para LLNs. Para poder utilizar o RPL e o IPV6 devem ser inseridas as seguintes linhas no Makefile: (1) $WITH\_UIP6 = 1$; (2) $UIP\_CONF\_IPV6 = 1$; (3) $CFLAGS+ = -DUIP\_CONF\_IPV6\_RPL$. Também foi observado que o tempo de simulação deve ser suficiente para a coleta de dados começar, visto que, os pacotes só começam a ser trocados após o DODAG ter sido criado.

Em nossas simulações, usamos um tempo virtual de 5 minutos. Usamos também 11 nós com uma faixa de transmissão de 50 metros e uma faixa de interferência de 100 metros. As medidas realizadas nos experimentos foram: Variação da taxa de serviço dos nós quando há ataque sem o sistema de proteção e quando há ataque com o sistema de proteção, impacto do ataque causado na rede e energia consumida no processo.

5.1 Simulação da rede sem presença de ataque

Depois de executar a simulação durante 5 minutos (virtuais), o Gráfico acíclico direcionado (DAGs) foi gerado e desenhado através de um dos *plug-ins* instalados no Cooja (*Sensor Map*) como mostrado na Figura 8. O *powertrace* foi o *plug-in* responsável por realizar a medição do consumo de energia de cada nó e o *collect-view* por coletar dados referentes aos nós após eles entrarem na rede. O *Sensor Map* é uma das opções encontradas dentro do *collect-view*. Para que o *Sensor Map* e o *powertrace* funcionem deve ser inserida a seguinte linha no Makefile: APPS = powertrace collect-view.

Figura 8 – O DODAG formado na rede simulada.

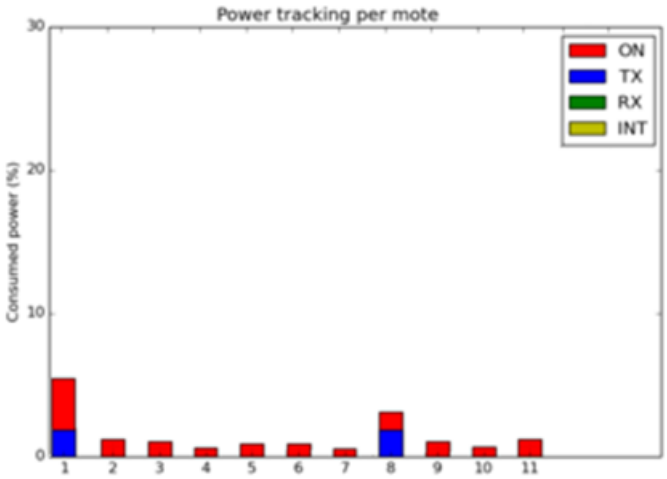


Fonte: Autoria própria

O nó com ID 1 é a raiz e o nó com ID 11 visto na Figura 7 e na Figura 8, primeiro é visto como um nó válido comum, permitindo a simulação da rede sem a presença de ataques. Para simular a presença dos ataques na rede, o nó com ID 11 foi modificado para atuar como o nó malicioso. Porém nesta primeira simulação, não temos ataques e todos os nós são válidos.

O consumo de energia, como já dito anteriormente, também pode ser medido através de um dos *plug-ins* instalados no Cooja (*powertrace*). O *powertrace* foi capaz de desenhar gráficos para ilustrar o gasto de cada um dos nós da rede. Como pode ser facilmente visto na Figura 9, todos os nós tiveram o consumo de energia inferior a 7 % do potencial total, sendo que a maior parte desse consumo ocorreu devido a formação do DODAG.

Figura 9 – Medição do consumo de energia da simulação sem a presença de ataques.



Fonte: Autoria própria

5.2 Simulação da rede com presença do ataque Blackhole (ou variante Selective Forward) unido ao Sinkhole

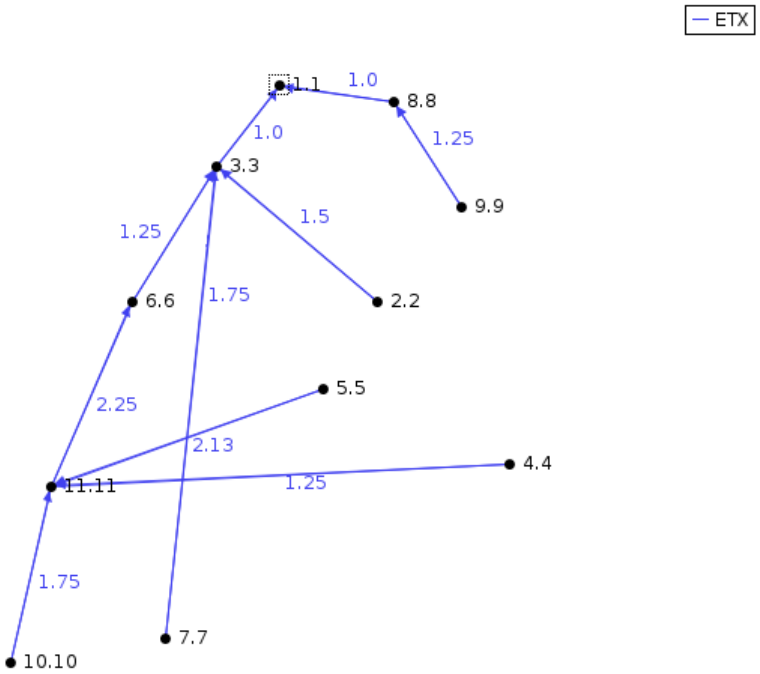
Nesta simulação, o nó com ID 11 passou a ser um nó malicioso. Para causar maior impacto à rede os ataques *Blackhole* e *Selective Forward* foram combinados com o ataque *Sinkhole*.

Ao simular o *Blackhole* e o *Selective Forward* combinado com *Sinkhole*, o DODAG foi alterado. Alguns nós válidos (ID 4, 5 e 10) na vizinhança do nó malicioso (ID 11) mudaram suas rotas escolhendo um novo pai (o nó malicioso). Desta forma, o ataque tornou-se ainda mais eficiente, pois passou a afetar um maior número de dispositivos, ouvindo e deixando cair um número maior de mensagens recebidas. O DODAG alterado pode ser visto na Figura 10 gerada pelo simulador Cooja.

No caso do *Blackhole* o atacante descartou todos os pacotes de mensagens coletados dos nós com ID 4, 5 e 10 em vez de encaminhá-los ao seu destino. Ao utilizar a variante desse

ataque conhecida como *Selective Forward* na simulação, apenas as mensagens do plano de dados recebidas do nó com IP especificado pelo nó invasor foram descartadas, o nó escolhido na simulação pelo atacante foi o nó com ID 5. Desta forma, foi facilmente observado um mau funcionamento na rede em relação à entrega de pacotes.

Figura 10 – O DODAG formado na rede simulada com ataque *Blackhole* (ou variante *Selective Forward*) unido ao ataque *Sinkhole*.



Fonte: Autoria própria

Para poder simular esse ataque foi necessário realizar algumas modificações nas constantes de configuração RPL, mais especificamente as alterações foram feitas nos arquivos "rpl-private.h" e "rpl-timers.c". Com essas modificações o nó malicioso anuncia uma classificação melhor do que os vizinhos, fazendo com que o DODAG seja modificado. Geralmente esses arquivos dentro do *ContikiOS* são encontrados no seguinte caminho: /home/user/net/rpl

Código 1 – Trecho de código original no arquivo rpl-private.h

```
1 #define RPL_MAX_RANKINC                    (7 * RPL_MIN_HOPRANKINC)
2 #define INFINITE_RANK                      0xffff
```

Código 2 – Trecho de código modificado no arquivo rpl-private.h

```
1 #define RPL_MAX_RANKINC                    0  /* VALOR ALTERADO */
2 #define INFINITE_RANK                      256   /* VALOR ALTERADO */
```

Código 3 – Trecho de código original no arquivo rpl-timers.c

```
static void
handle_periodic_timer(void *ptr)
{
  rpl_purge_routes();
  rpl_recalculate_ranks();
  /* handle DIS */
#if RPL_DIS_SEND
          next_dis++;
          if(rpl_get_any_dag() == NULL && next_dis >= RPL_DIS_INTERVAL) {
                    next_dis = 0;
                    dis_output(NULL);
          }
#endif
 ctimer_reset(&periodic_timer);
}
```

Código 4 – Trecho de código modificado no arquivo rpl-timers.c

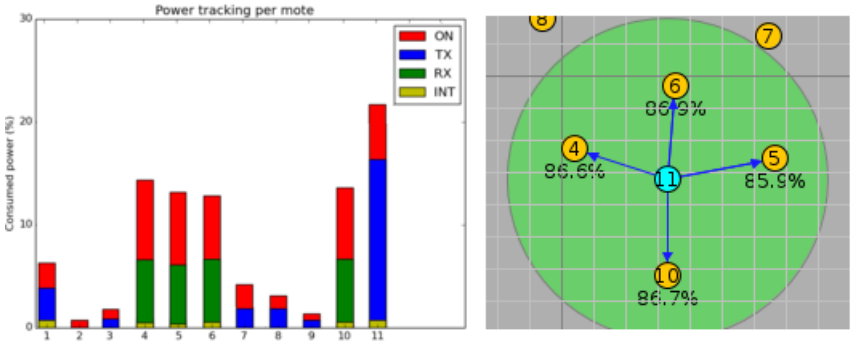
```
static void
handle_periodic_timer(void *ptr)
{
  rpl_purge_routes();
  /*  rpl_recalculate_ranks(); ESSA LINHA DEVE SER COMENTADA */
  /* handle DIS */
#if RPL_DIS_SEND
          next_dis++;
          if(rpl_get_any_dag() == NULL && next_dis >= RPL_DIS_INTERVAL) {
                    next_dis = 0;
                    dis_output(NULL);
          }
#endif
 ctimer_reset(&periodic_timer);
}
```

5.3 Simulação da rede com presença do ataque Flooding

Nesta simulação, o nó malicioso (com ID 11) afeta os nós com IDs 4, 5, 6 e 10. É possível perceber ao observar a Figura 11 que esses nós são particularmente afetados pelo ataque em termos de RX (Escuta). Além disso, é notório que o atacante consome bastante energia em termos de TX (Transmissão). Portanto, os nós afetados pelo ataque gastam muita energia e

memória para tentar ler as inúmeras solicitações de conexão enviadas pelo atacante. A energia consumida pelo nó atacante e pelos nós afetados pelo ataque é superior aos 7% do valor total que constava como limite para os nós na simulação sem a presença de ataque. O consumo chegou a mais de 20% no caso do atacante e a 14% no caso do nó mais afetado pelo ataque. Sendo assim o nó mais afetado chegou a consumir mais do que o dobro de energia se comparado o seu consumo na presença do ataque ao seu consumo na simulação sem a presença de ataque, vista na Figura 9. A energia consumida pelos nós não afetados permaneceram abaixo de 7%.

Figura 11 – Medição do consumo de energia da simulação com a presença do ataque *Flooding*.



Fonte: Autoria própria

Para poder simular o ataque *Flooding* foi necessário realizar algumas modificações nas constantes de configuração RPL, mais especificamente as alterações foram feitas no arquivo "rpl-timers.c". Com essas modificações o nó malicioso irá realizar DoS (negação de serviço) ao preencher o *buffer* de memória dos nós afetados enviando uma quantidade maior do que o normal de solicitações de conexão.

Código 5 – Trecho de código original no arquivo rpl-timers.c

```
1  static void
2  handle_periodic_timer(void *ptr)
3  {
4    rpl_purge_routes();
5    rpl_recalculate_ranks();
6    /* handle DIS */
7  #if RPL_DIS_SEND
8    next_dis++;
9    if(rpl_get_any_dag() == NULL && next_dis >= RPL_DIS_INTERVAL) {
10     next_dis = 0;
11     dis_output(NULL);
12   }
13 #endif
14   ctimer_reset(&periodic_timer);
```

Código 6 – Trecho de código modificado no arquivo rpl-timers.c

```
static void
handle_periodic_timer(void *ptr)
{
  rpl_purge_routes();
  rpl_recalculate_ranks();
  /* handle DIS */
#if RPL_DIS_SEND
  next_dis++;
  int i=0;
  while (i<10) {
    i++;
    dis_output(NULL);
  }
  if(rpl_get_any_dag() == NULL && next_dis >= RPL_DIS_INTERVAL) {
    next_dis = 0;
    dis_output(NULL);
  }
#endif
  ctimer_reset(&periodic_timer);
}
```

5.4 Identificação do Ataque e do nó atacante

Para construir a lógica fuzzy responsável pela identificação do ataque e atacante *Flooding* consideramos que X fosse o número de pacotes enviados de cada nó, MÍNIMO fosse o número de pacotes enviados pelo nó que menos efetuou envio de pacotes e MÉDIA fosse a média de pacotes enviados por todos os nós da rede. Para escolher o valor de MÁXIMO foi preciso realizar algumas simulações do ataque *Flooding* com o atacante enviando um número de pacotes mínimo necessário para causar danos à rede e ser detectado pelo IDS, esse número foi aprendido e utilizado no valor de MÁXIMO.

Para construir a lógica fuzzy responsável pela identificação do ataque e dos nós que mais sofreram os efeitos do ataque *Blackhole* ou *Selective Forward* consideramos que X fosse o número de pacotes perdidos de cada nó, MÍNIMO fosse o número de pacotes perdidos pelo nó que menos perdeu pacotes e MÉDIA seja a média de pacotes perdidos por todos os nós da rede. Para escolher o valor de MÁXIMO foi preciso realizar algumas simulações do ataque *Blackhole* e do *Selective Forward* com os nós afetados pelo ataque tendo um número mínimo de pacotes descartados necessários para causar danos à rede e ser detectado o ataque pelo IDS, esse número mínimo de pacotes descartados foi aprendido e utilizado no valor de MÁXIMO.

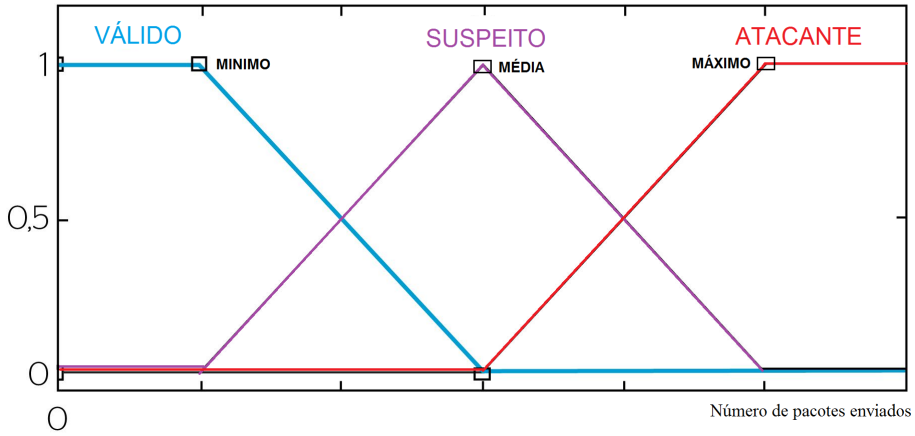
Esses valores ficam guardados para futuras detecções e podem ser atualizados a medida

que forem surgindo valores que devam substituir o MÁXIMO, a MÉDIA e o MÍNIMO. Outros dados necessários foram definidos da seguinte forma:

- Como variáveis de entrada, foram levados em consideração os valores de X, MINIMO, MÉDIA e MÁXIMO;
- As variáveis linguísticas relacionadas aos Nós da rede foram VÁLIDO, SUSPEITO e ATACANTE no caso do *Flooding* ou ATACADO no caso do *Blackhole* e *Selective Forward*. A variável linguística relacionada à presença do nó na Lista Negra foi AUSENTE;
- Os intervalos ficaram definidos da seguinte forma:
  - Para VALIDO, foi atribuída total pertinência $\forall(X)$ entre [0, MÍNIMO];
  - Para SUSPEITO, foi atribuída total pertinência $\forall(X)$ = MÉDIA;
  - Para ATACANTE ou ATACADO, foi atribuída total pertinência $\forall(X)$ >= MÁXIMO.
  - Para AUSENTE, foi aplicada a lógica binária comum onde o nó está totalmente ausente da Lista negra (1) ou não (0), sem haver intermediários valores de verdade.

  Ao observar o gráfico ilustrado na Figura 12 pode se obter melhor entendimento na escolha dos intervalos de valores escolhidos.

Figura 12 – Gráfico da nossa função de pertinência fuzzy



Fonte: Autoria própria

- Houve a transferência da Base de Conhecimento do ambiente de desenvolvimento para o controlador fuzzy. Nesse momento o controlador fuzzy verifica os valores referentes as variáveis linguísticas para cada nó da rede;
- A base de regras ficou definida da seguinte forma:
  - Se o Nó for VALIDO e AUSENTE então o nó é comum;

  - Se o Nó for SUSPEITO e AUSENTE então o nó é mais ou menos atacante;
  - Se o Nó for ATACANTE e AUSENTE então o nó é um provável atacante;

- Para realizar a Defuzzyficação (D) dos valores nebulosos de saída da base de regras foi aplicada uma fórmula, nesta fórmula $R_a(X)$ corresponde a função de pertinência que será sombreada no gráfico visto na figura 12. O valor encontrado após aplicar a deffuzzyficação é o grau de potencial do tipo de ataque ocorrido e do nó ser classificado como atacante (no caso do *Flooding*) ou atacado (no caso do *Selective Forward*). Caso o ataque mais provável seja o *Flooding* o nó que possuir maior grau de potencial de ser atacante terá o valor da variável linguística AUSENTE alterado para 0 tornando-se dessa forma um dos nós presentes na Lista Negra. Caso o ataque mais provável seja o *Selective Forward* o nó que possuir o maior grau de potencial de ser o atacado terá o seu pai do DODAG, adicionado à Lista Negra alterando o valor da variável linguística AUSENTE referente a ele para 0. A formula mencionada para calcular o valor de D pode ser vista em seguida:

$$D = \frac{\sum R_a(X) * X}{(\sum R_a(X))} \tag{5.1}$$

Ao analisar o sistema percebeu-se que com essas variáveis e intervalos é possível detectar o nó atacante ao verificar o valor de verdade para cada uma das variáveis linguísticas. Uma outra observação a ser destacada é que a taxa de acerto e falsos positivos estão diretamente relacionadas ao tempo que o atacante permaneceu atuando na rede. Isso porque quanto mais dados fora do comum coletados pelo módulo de monitoramento e analisados pelo módulo de análise existir, maior a probabilidade de acerto ao tentar detectar o nó malicioso. O gasto de energia referente ao módulo de planejamento concentra-se apenas no 6BR aumentando em cerca de 30% o seu consumo.

5.5 Simulação da rede ao isolar o nó malicioso

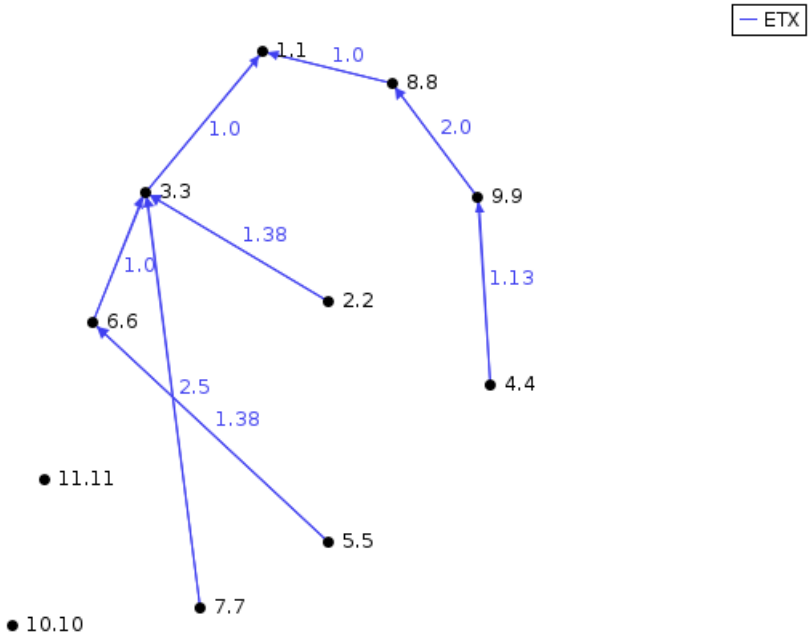
Durante a fase de execução, nossa arquitetura destina-se a isolar o nó do atacante para que não cause mais danos à rede. Ao isolar o nó houve uma melhora significante no funcionamento da rede tanto no consumo de energia considerado a longo prazo, quanto na taxa de serviço relacionada a tramitação de pacotes, já que como o nó atacante não estava mais presente na rede não houveram perdas de pacotes relacionadas a ataque. O benefício relacionado ao consumo de energia é considerado a longo prazo visto que, a reconstrução do DODAG a partir do zero consome energia de todos os nós da rede devido a troca de mensagens DIS, DIO e DAO porém os nós afetados pelos ataques terão a segurança de voltar a participar da rede sem sofrer danos causados pelo atacante.

No reparo global comum do RPL os nós iniciam a troca de mensagens DIS, DIO e DAO até que que o DODAG seja totalmente reconstruído, já no reparo global modificado do RPL os

nós válidos ignoram os DIS e DIO dos nós presentes na Lista negra (Aqueles que possuem o valor da variável linguística fuzzy AUSENTE com o valor 0) neste caso o nó com ID 11. Como o reparo global é iniciado a partir da raiz (6BR), o 6BR além de ignorar os nós presentes na Lista Negra é responsável por informar aos nós válidos que forem ingressando a rede quem são os nós classificados como atacantes.

Em um dos cenários simulados ao isolar o nó atacante, o DODAG foi recriado e outro nó, além do nó malicioso também foi isolado. O nó isolado indevidamente, pode ser visto na Figura 13. Isso ocorreu porque o nó com ID 10 estava muito longe dos outros, excedendo o limite máximo de 50 metros da faixa de transferência utilizada na simulação. Ao diminuir a distância dos nós válidos com o nó de ID 10 para um valor inferior a 50 metros foi possível criar uma nova rota possibilitando que o nó isolado indevidamente voltasse a pertencer a rede.

Figura 13 – O DODAG da rede simulada após isolar nó malicioso



Fonte: Autoria própria

6
Conclusão

Neste trabalho, utilizamos o laço de controle MAPE-K para projetar uma nova arquitetura de autoproteção para a IoT. Com esta arquitetura, é possível incorporar instalações de segurança nos sistemas IoT que liberam o programador de algumas tarefas necessárias para o funcionamento seguro da rede. Ao utilizar a arquitetura funções como monitoramento da rede, detecção de ataque, classificação do ataque e ações voltadas a mitigar os danos causados à rede tornaram-se automatizadas. Além disso, o usuário pode ampliar a arquitetura desenvolvendo novos serviços de segurança para lidar com ataques específicos.

A arquitetura ajuda a fornecer um mecanismo de autoproteção para as redes da IoT, garantindo um melhor desempenho e aumentando a confiança dos usuários ao usar dispositivos conectados nesse ambiente. Na presente dissertação foram focados os módulos de planejamento e execução.

No módulo de planejamento para classificar o ataque e o nó atacante foi utilizada a lógica fuzzy. No módulo de execução foram realizadas algumas modificações no RPL com intuito de acabar com o efeito do ataque sobre a rede.

A nossa arquitetura trata de quatro ataques diferentes (*Sinkhole, Selective Forward, Blackhole e Flooding*) tendo em mente o impacto causado no funcionamento da rede devido à falta de recursos dos dispositivos disponíveis no ambiente da IoT. Esses cinco ataques foram implementados e simulados através do Cooja, simulador do ContikiOS, expondo o consumo de energia e a formação do DODAG RPL.

6.1 Trabalhos Futuros

O desempenho do sistema foi avaliado verificando o impacto dos ataques à rede e se a arquitetura de autoproteção ameniza os danos causados por eles. Porém novos ataques e novas

tecnologias surgirão, então o trabalho aqui apresentado poderá ser estendido para abordar-los.

Além das técnicas apresentadas é possível acoplar novas técnicas, substituir-las ou até mesmo aperfeiçoa-las como no caso da lógica fuzzy onde poderão ser utilizados outros pesos, outros intervalos de valores e uma diferente base de regras, tudo isso utilizando a mesma arquitetura. Realizar testes com técnicas diferentes é importante para que sejam utilizadas aquelas trouxerem melhores resultados para o problema abordado.

Ainda existem novos cenários de testes a serem simulados com intuito de se preparar para toda e qualquer possível situação fora do comum que venha ocorrer à rede IoT. Um exemplo de um desses novos cenários, seria um ambiente simulado com a presença de diversos atacantes realizando ataques distintos.

Referências

ALMEIDA, F. M. de; RIBEIRO, A. d. R. L.; MORENO, E. D. An architecture for self-healing in internet of things. *UBICOMM 2015*, p. 89, 2015. Citado 3 vezes nas páginas 16, 17 e 42.

ASHTON, K. That 'internet of things' thing. *RFiD Journal*, v. 22, n. 7, 2011. Citado na página 19.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer networks*, Elsevier, v. 54, p. 2787–2805, October 2010. Doi:10.1016/j.comnet.2010.05.010. Citado 2 vezes nas páginas 14 e 30.

CERVANTES, C. et al. Detection of sinkhole attacks for supporting secure routing on 6lowpan for internet of things. In: IEEE. *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. [S.l.], 2015. p. 606–611. Citado 2 vezes nas páginas 37 e 40.

CULLER, D.; CHAKRABARTI, S. 6lowpan: Incorporating ieee 802.15. 4 into the ip architecture. *IPSO Alliance White Paper*, 2009. Citado 2 vezes nas páginas 24 e 25.

DOBSON, S. et al. Fulfilling the vision of autonomic computing. *IEEE Computer*, January 2010. Doi:10.1109/MC.2010.14. Citado 3 vezes nas páginas 14, 20 e 21.

DUNKELS, A.; GRONVALL, B.; VOIGT, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In: IEEE. *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. [S.l.], 2004. p. 455–462. Citado 2 vezes nas páginas 22 e 23.

GOMIDE, F.; ROCHA, A.; ALBERTOS, P. Neurofuzzy controllers-ifac-lca'92. *Viena, Austria*, 1992. Citado na página 36.

GOMIDE, F. A. C.; GUDWIN, R. R. Modelagem, controle, sistemas e lógica fuzzy. *SBA Controle & Automação*, v. 4, n. 3, p. 97–115, 1994. Citado 3 vezes nas páginas 34, 35 e 36.

GOYAL, P.; BATRA, S.; SINGH, A. A literature review of security attack in mobile ad-hoc networks. *International Journal of Computer Applications*, Citeseer, v. 9, p. 11–15, November 2010. Citado na página 31.

GREENSMITH, J.; AICKELIN, U.; CAYZER, S. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In: *Artificial Immune Systems*. [S.l.]: Springer, 2005. p. 153–167. Doi:10.1007/11536444_12. Citado na página 30.

GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, Elsevier, v. 29, p. 1645–1660, September 2013. Doi:10.1016/j.future.2013.01.010. Citado 3 vezes nas páginas 14, 15 e 20.

HEER, T. et al. Security challenges in the ip-based internet of things. *Wireless Personal Communications*, Springer, v. 61, p. 527–542, September 2011. Doi:10.1007/s11277-011-0385-5. Citado na página 32.

HUI, J. et al. An ipv6 routing header for source routes with rpl. *draft-ietf-6man-rpl-routing-header-00, IETF*, 2010. Citado na página 28.

KARLOF, C.; WAGNER, D. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, Elsevier, v. 1, p. 293–315, September 2003. Citado na página 32.

KEPHART, J. et al. Blueprint for a computer immune system. In: *Artificial immune systems and their applications*. [S.l.]: Springer, 1999. p. 242–261. Citado na página 42.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. *Computer*, IEEE, v. 36, p. 41–50, January 2003. Doi:10.1109/MC.2003.1160055. Citado 3 vezes nas páginas 15, 20 e 21.

LEE, C.-C. Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on systems, man, and cybernetics*, IEEE, v. 20, n. 2, p. 404–418, 1990. Citado na página 33.

LIU, C. et al. Research on immunity-based intrusion detection technology for the internet of things. In: IEEE. *Natural Computation (ICNC), 2011 Seventh International Conference on*. [S.l.], 2011. v. 1, p. 212–216. Citado 3 vezes nas páginas 37, 39 e 41.

MARTINS, D.; GUYENNET, H. Wireless sensor network attacks and security mechanisms: A short survey. In: IEEE. *Network-Based Information Systems (NBiS), 2010 13th International Conference on*. [S.l.], 2010. p. 313–320. Doi:10.1109/NBiS.2010.11. Citado 2 vezes nas páginas 30 e 31.

MONTENEGRO, G. et al. *Transmission of IPv6 packets over IEEE 802.15. 4 networks*. [S.l.], 2007. Citado 2 vezes nas páginas 24 e 25.

NAMI, M. R.; SHARIFI, M. Autonomic computing: a new approach. In: IEEE. *Modelling & Simulation, 2007. AMS'07. First Asia International Conference on*. [S.l.], 2007. p. 352–357. Citado na página 15.

NARTEN, T. et al. Neighbor discovery for ip version 6 (ipv6). 2007. Citado na página 24.

OSTERLIND, F. A sensor network simulator for the contiki os. *Swedish Institute of Computer Science (SICS), Tech. Rep. T2006-05*, 2006. Citado 2 vezes nas páginas 23 e 24.

RAZA, S.; WALLGREN, L.; VOIGT, T. Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, Elsevier, v. 11, p. 2661–2674, November 2013. Citado 2 vezes nas páginas 37 e 38.

REUSING, T. Comparison of operating systems tinyos and contiki. *Sens. Nodes-Operation, Netw. Appli.(SN)*, v. 7, 2012. Citado 3 vezes nas páginas 21, 22 e 23.

RODRÍGUEZ, R. M. et al. Hesitant fuzzy sets: state of the art and future directions. *International Journal of Intelligent Systems*, Wiley Online Library, v. 29, n. 6, p. 495–524, 2014. Citado na página 45.

ROMAN, R.; NAJERA, P.; LOPEZ, J. Securing the internet of things. *Computer*, IEEE, v. 44, p. 51–58, September 2011. Doi:10.1109/MC.2011.291. Citado 4 vezes nas páginas 14, 15, 19 e 20.

SHILA, D. M.; CHENG, Y.; ANJALI, T. Mitigating selective forwarding attacks with a channel-aware approach in wmns. *Wireless Communications, IEEE Transactions on*, IEEE, v. 9, p. 1661–1675, May 2010. Doi:10.1109/TWC.2010.05.090700. Citado 4 vezes nas páginas 30, 31, 37 e 38.

VASSEUR, J. et al. Rpl: The ip routing protocol designed for low power and lossy networks. *Internet Protocol for Smart Objects (IPSO) Alliance*, v. 36, 2011. Citado 2 vezes nas páginas 25 e 26.

WALLGREN, L.; RAZA, S.; VOIGT, T. Routing attacks and countermeasures in the rpl-based internet of things. *International Journal of Distributed Sensor Networks*, v. 2013, June 2013. Citado na página 31.

WANG, X.; ZHONG, S.; ZHOU, R. A mobility support scheme for 6lowpan. *Computer Communications*, Elsevier, v. 35, n. 3, p. 392–404, 2012. Citado 3 vezes nas páginas 20, 24 e 25.

WEYNS, D.; MALEK, S.; ANDERSSON, J. Forms: a formal reference model for self-adaptation. In: ACM. *Proceedings of the 7th international conference on Autonomic computing*. [S.l.], 2010. p. 205–214. Doi:10.1145/1809049.1809078. Citado na página 22.

WINTER, T. Rpl: Ipv6 routing protocol for low-power and lossy networks. 2012. Citado 3 vezes nas páginas 25, 26 e 27.

WOOD, A. D.; STANKOVIC, J. A. Denial of service in sensor networks. *Computer*, IEEE, v. 35, p. 54–62, December 2002. Doi:10.1109/MC.2002.1039518. Citado 2 vezes nas páginas 31 e 32.

XU, L. D.; HE, W.; LI, S. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, IEEE, v. 10, n. 4, p. 2233–2243, 2014. Citado 2 vezes nas páginas 15 e 19.

ZADEH, L. A. Fuzzy sets. *Information and control*, Elsevier, v. 8, n. 3, p. 338–353, 1965. Citado na página 32.

ZADEH, L. A. Fuzzy logic= computing with words. *IEEE transactions on fuzzy systems*, IEEE, v. 4, n. 2, p. 103–111, 1996. Citado 4 vezes nas páginas 32, 33, 35 e 36.

Apêndices

APÊNDICE A – AICT-B1

Mitigating Attacks in the Internet of Things with a Self-protecting Architecture

Ruan de A. C. Mello, Admilson de R. L. Ribeiro, Fernando M. de Almeida, Edward D. Moreno
Department of Computing
Federal University Sergipe UFS
São Cristóvão, Brazil
E-mails: ruanmello@gmail.com, admilson@ufs.br, fernando.m.al.91@gmail.com, edwdavid@gmail.com

***Abstract*—Internet of Things (IoT) applications run in environments that have resource constraints and are unsecured. Due to the nature of their environment, IoT systems should be able to reason autonomously and take self-protecting decisions. Currently, an adequate architecture to incorporate self-protection in the IoT is not available. Thus, we design a new self-protecting architecture based on the MAPE-K (Monitor, Analyze, Plan, Execute and Knowledge) autonomic control loop that will run at the application layer so that developers can add several security services. In this paper, we address the impact caused by attacks (SinkHole, Selective Forward, Black Hole and Flooding) in relation to power consumption and interference in the operation of the network created from the Routing Protocol for Low Power and Lossy Networks (RPL) routing protocol.**

Keywords–Autonomic Systems; Self-protecting; IoT; MAPE-K loop; AIS.

I. INTRODUCTION

Recently, the integration of embedded systems, wireless networks and the Internet led to a new application type, namely Internet of Things (IoT) applications. A particular case of IoT applications is the participatory sensing application where people that live in communities and are dependent on each other for daily activities exchange information to reach their objectives [1]. Recommendations for a good restaurant, car mechanic, movie, phone plan and so on were and still are some things where community knowledge helps us in determining our actions.

IoT applications will have a great impact on people's life, but currently only a small number of such applications is available to our society. As the things are nodes of a network, individuals can control, locate, and monitor everyday objects remotely. For example, the use of wireless sensor technologies allows monitoring the health of people in real time, enabling brief diagnostics. The vital parameters of individuals such as blood pressure, temperature, and so on, are measured through sensor nodes that stay on the bodies of patients that continue to do their daily activities [2]. Many benefits can be provided by the IoT technologies in the health-care domain.

However, IoT applications run in environments that have resource constraints and are unsecured. The resources constraint of the IoT devices can lead to security breaches. For example, an attacker can try to maintain the IoT devices in operation all the time, with the intention to consume all their battery energy. This attack is a type of Denial of Service (DoS) attack and can have a great impact on the application availability without the possibility of control by users. Therefore, due to that environment, the security issue must be treated autonomously. That is, the self-protection property must be incorporated in the IoT systems [3].

However, the majority of security mechanisms in IoT is composed of protocols and algorithms that run at the physical layer or link layer of the protocol stack of the software system [4]. These mechanisms are adequate to protect against the problems relating to the confidentiality and the integrity, but in some cases they fail on considering the availability.

Considering the availability of applications, self-protection is the essential property that allows network nodes to communicate and react to attacks of hackers according to security policies defined by users [5]. Thus, IoT systems should be able to reason autonomously and make self-protecting decisions.

Therefore, in this context, we propose a self-protecting architecture for the Internet of Things based on the MAPE-K control loop [5] and the danger theory of the Artificial Intelligent System (AIS) [6]. To show the use of the architecture, we implement the execution phase describing the main attacks in the IoT and their impacts in relation to power consumption and interference in the operation of the network.

The remainder of this paper is organized into nine more sections. Section II presents the limitations of related works and highlights our contribution. Section III presents the autonomic loop MAPE-K. Section IV presents the Routing Protocol for Low Power and Lossy Networks. Section V gives a brief overview of the main attacks that occur in the IoT environment. Section VI outlines our architecture, considering the MAPE-K control loop and its phases are described. Section VII discusses how we implement the execution phase of our architecture. Section VIII presents the results obtained so far. Section IX concludes the paper and presents future works.

II. RELATED WORK

In the literature, there are several papers about computing security based on the AISs and autonomic computing. Kephart et al. [6] and White et al. [7] designed the first AISs in response to the first virus epidemics, when it was found out that the spreading of cure had to be faster that the contamination by viruses. After, SweetBair [8] used a more sophisticated technique to capture suspecting traffic and generate signatures of worms. As a variant of this pattern, Swimmer [9] and also Rawat and Saxena [10] presented an approach based on danger theory for attack detection in autonomic networks.

SVELTE [11], as the authors claim, is the first Intrusion Detector System (IDS) for the IoT. The work presented has a huge contribution to design an IDS with the characteristics of a network for IoT, considering the technologies used in

the communications stack, such as IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) and RPL routing protocol. However, its approach does not have autonomic characteristics for self protect the network from further attacks, only the determined attack on the network project level. Like the SVELT, the CAD [12] not only detects the attack, but also tries to mitigate the damage caused by the attacker. The CAD is directed to the wireless mesh network (WMN) and can differentiate between losses occurring in the normal events of a legitimate attack of the type Selective Forward.

The main limitation is that they are not well suited for the IoT environment. They only present some particular solutions that address a set of specific problems different of the IoT environment. Besides, they do not consider the possibility of development of new self-protecting services. Therefore, programmers are unable to decide which policies are more appropriate for their applications, considering still that the resolved policies of low-level protocols sometimes are not the most appropriate for all applications in the IoT.

Therefore, our solution advances previous solutions providing a new self-protecting architecture for the IoT and also the possibility to extend such architecture with new security services.

III. MAPE-K AUTONOMIC CONTROL LOOP

In March 2001, Paul Horn presented for the first time the MAPE-K Autonomic Control Loop at an IBM event. The MAPE-K Loop was presented as a reference model. Composed of five modules that can be seen in Figure 1, the MAPEK-K Loop is intended to distribute the tasks of each element of the autonomic computing [13]. The modules that build the MAPE-K Loop are, respectively, knowledge, monitoring, analysis, planning, and execution.

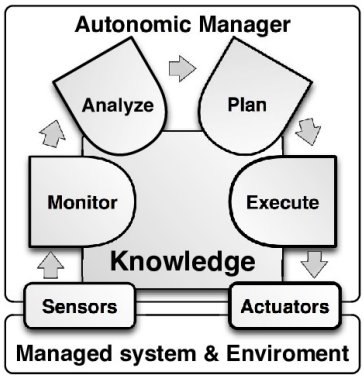


Figure 1. Mape-K Autonomic Control Loop [14].

- The Knowledge module is in charge of keeping relevant data in the memory to accelerate decision making.
- The Monitoring module uses sensors to collect data from the managed element, which could be a software or hardware resource, or an autonomic manager itself.
- The Analysis module provides mechanisms to interpret the collected data from the monitoring phase and predict future situations.
- The Planning module builds the necessary actions to achieve the goals.
- The Execution module uses effects to make changes on managed elements.

IV. ROUTING PROTOCOL FOR LOW POWER AND LOSSY NETWORKS

RPL is an IPv6 routing protocol for Low power and Lossy Networks (LLNs) that specifies how to build a Destination Oriented Directed Acyclic Graph (DODAG) using an objective function and a set of metrics and constraints to compute the best path [15].

The RPL differs from other routing protocols that operate in less-constrained environments. In LLNs, especially when the network is made of devices that must save energy, it is imperative to limit the control plane traffic in the network.

The graph built by RPL is a logical routing topology built over a physical network to meet a specific criteria and the network administrator may decide to have multiple routing topologies (graphs) active at the same time used to carry traffic with different set of requirements [15].

V. ATTACKS IN INTERNET OF THINGS

Most of threats in IoT environment attack the limited power of the sensors. The limited power of these devices exposes the network to many threats [16]. In this section, we will discuss some of the latest and more common attacks on the environment of the IoT and wireless sensor networks [17].

1) Selective forward: In a Selective Forward attack, the attacker node receives the transmission packets, but refuses to transmit some of them and drops those that it refused to transmit. The attacker must choose which packets to discard according to some standard such as size, destination or origin [12]. In this case, only the packets released by the attacker node can be freely transmitted.

2) Black Hole: In a Black Hole attack, the attacker node receives the transmission packets and drops all packets received, regardless of type, size, origin or destination [12].

3) Sinkhole: In a Sinkhole attack, the attacker tries to attract all the traffic from neighboring nodes [18]. So, practically, the attacker node listens to all data transmitted from neighboring nodes. Only this attack does not cause too much damage in the network, but together with another type of attack (Selective Forward or Black Hole), can become very powerful.

4) Flooding: In a Flooding attack, the attacker explores the vulnerabilities related to the depletion of the memory and the energy. One manner to take advantage of this vulnerability is when an opponent sends too many requests trying to connect to the victim, every request makes the victim allocate the resources in an attempt to maintain the connection [19]. Thus, to prevent the total resource depletion is necessary to limit the number of connections. However, this solution also prevents valid nodes to create a connection with the victim, causing problems such as queuing [19].

5) Hello Flood: In a HelloFlood attack, the attacker uses a device with a powerful signal to regularly send some messages; that way, the network is left in a state of confusion [17]. In order to find ad-hoc networks, many protocols use Hello Messages for discovering neighbor nodes and automatically

create a network. With the Hello Flood attack, an attacker can use a device with high transmission power to convince every other node in the network that the attacker is its neighbor, but these nodes are far away from the attacker. In this case, the power consumption of sensors is significantly increased, because of protocols that depend on exchange information between neighbor nodes for topology maintenance or flow control [17].

Previously, we saw some of the most common attacks on IoT networks and, next, we analyzed the possible strategies to end or to mitigate the damage caused by them.

To stop the damages caused by attacks on a network, first it is necessary to detect these attacks, using an IDS. An IDS analyzes network activity and attempts to detect any unusual behavior that may affect the integrity of the network. Based on information provided by IDS, strategies are created to cope with the attacks. For example:

- To mitigate Sinkhole - If the geographical locations of the nodes of RPL DODAG are known, the effect of Sinkhole attacks can be mitigated by the use of flow control, making sure, that the messages are traveling to the correct destination. The RPL protocol also supports multiple instances DODAG offering alternative routes to the root DODAG [20].
- To mitigate Hello Flood - A simple solution to this attack, it is perform a bidirectional check for each message "HELLO" [21]. If there is no recognition, the path is assumed to be bad and a different route is chosen. If geographical locations of the nodes of RPL DODAG are known, all packets received from a node that is far beyond of the common network node transmission capacity can be dropped.
- To mitigate Selective Forward - An effective countermeasure against Selective Forward attacks is to ensure that the attacker cannot distinguish the different type of packets, forcing the attacker to send all or none packets [22].

Raza et al. [11] indicated that the most efficient and fastest way to stop the damage of routing attacks is to isolate the malicious node. Some forms to ignore the attacker node were studied. These forms are:

- The Black List: After identifying the nodes and finding the attackers, a list will be created and all the malicious nodes will be added in order to exclude them from the possible routes of traffic data. To ignore the attacker, a verification will be done against the Black List excluding all nodes found of the typical RPL DODAG that have a root and multiple nodes.
- The Gray List: After identifying the nodes and finding the attacker, a list will be created. The suspicious attacker node will be added to this list with the intention of excluding it from the possible routes of traffic data, for a predetermined time. After the end of the predetermined time the suspicious attacker node is deleted from the list. In this way, if there is any doubt about the identity of the attacker node, the node may re-join the network. To ignore the suspicious attacker nodes, when creating the routing, a verification will be done against the Gray List excluding all nodes found of the typical RPL DODAG that have a root and multiple nodes.
- The White List: As in the example of the Black List, a list will be created after identifying the nodes and finding the attacker node. But, this time, will be add into the White List only the valid nodes and all malicious nodes will be excluded. This way will have a verification stating which nodes are valid and must belong to a typical RPL DODAG with a root and several nodes.

VI. SELF-PROTECTION ARCHITECTURE

In Figure 2, we can see the self-protecting architecture for IoT proposed in this research. It consists of five modules (Monitoring, Analysis, Planning, Executing and Knowledge) and it is based on the MAPE-K autonomic control loop [13].

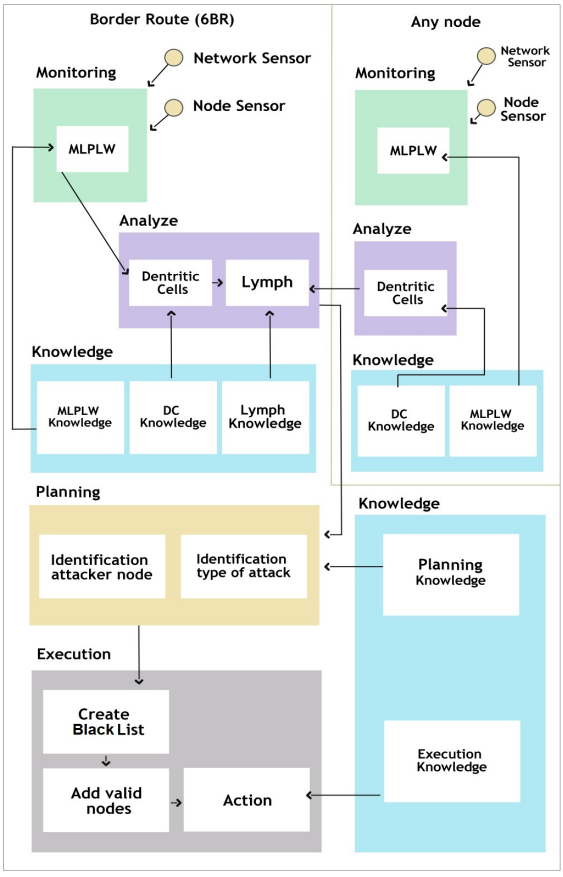


Figure 2. The Self-Protecting Architecture.

The monitoring and analysis modules are responsible, respectively, for collecting through the sensors, some information of the network that will be analyzed to measure the possibility of being associated with an attack. These two modules are present in the network nodes and in the border router (6BR).

The planning and execution modules will be responsible, respectively, for identifying the attacker, the type of attack and to mitigate the damage in the network. The information listed as relevant data for analysis, planning and execution is: type of transport protocol, type of application protocol, time of communication, number of messages sent, number of messages effectively sent and number of messages received.

The components of knowledge phase will be responsible for keeping all the knowledge acquired by the system. Knowledge about the planning and execution modules will be at 6BR. The information kept by the Knowledge Module will be used to facilitate and accelerate the discovery of the type of attack, the attacker node and the action to be taken to protect the network.

The complete design of this architecture can be found in Mello et al. [23]. The monitoring and analysis phases were implemented in [24]. Now, we describe how we implement the execution phase.

VII. Execution Phase

The component responsible for the execution phase should mitigate or stop the damage caused by the attacks occurred on the network. The type of attack and the identification of the attacker node will be the information that will influence in the choice of the predetermined action to mitigate or stop the damage in the network. These two important pieces of information will be provided by the Planning Phase. The reason to find out the attacker node is, trying to isolate as quickly as possible and create a new route, thus, avoiding further damage to the network. The type of attack will be among one of the two groups mentioned in Figure 3. According to the group selected there will be a specific action to solve the problem, because each type of attack causes different types of damage on the network.

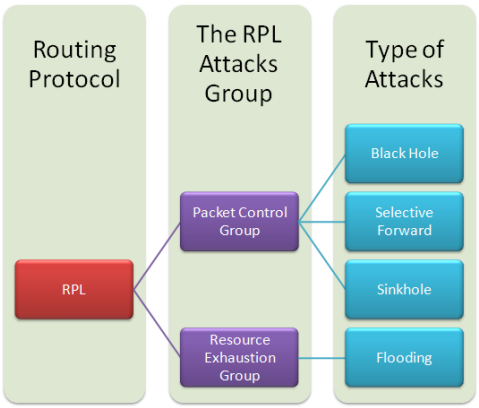


Figure 3. Taxonomy of RPL attacks

The first action to be taken by the components of the execution module, after the attack is detected, it is to ignore the malicious node. To perform this action it is important to identify the network nodes as legitimate or malicious. Raza et al. [11] say that it is necessary to be careful with the way of identifying nodes. If possible, it should be avoided the identification by IP address or MAC address, because they can be easily falsified. After making the identification of the nodes, some ways to ignore the attacker node were studied. Three ways to isolate the malicious node were analyzed before choosing the most convenient. The three ways are: Black List, Gray List and White List.

The way chosen was the Black List, because the maintenance of this list is simple. This way, all valid nodes will recalculate their rank in the RPL protocol (DODAG). To recalculate the rank of all valid nodes, it will be necessary to ignore the DODAG Information Object (DIO) of all nodes with higher rank than theirs and the DODAG Information Solicitation (DIS) and DIO of nodes that are present in the Black List. Thus, for a stranger node to join the network, it should be reported as safe and not be present in the Black List.

VIII. Results

It was possible to simulate the Flooding and Black Hole attacks (with SinkHole and Selective Forward variants). The simulations with and without the attacking node were performed in the Cooja, a simulator of the ContikiOS, following a DODAG topology in which there is a certain number of nodes and one of them will be the root.

We defined the number of nodes in the simulation and all used the same platform (Skymote). The routing protocol used was the RPL and the addressing protocol used was the IPV6.

It should also be noted that the simulation time should be long enough for the data collection to begin. In our simulation, we used a virtual time of 2 minutes. In the simulations, we used 11 nodes with a transmission rate of 50 meters and interference range of 100 meters. The network simulation was generated from the Cooja Simulator and can be seen in Figure 4.

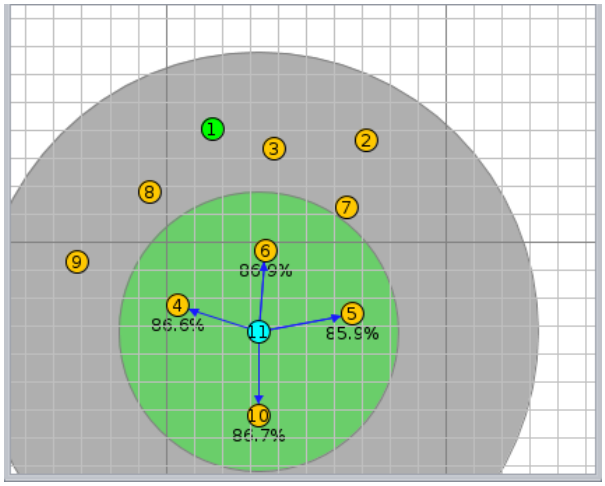


Figure 4. Network Simulation.

After running the simulation for 2 minutes (virtual), the Directed Acyclic Graph (DAGs) was generated by the Cooja Simulator looks as shown in Figure 5. The node with ID 1 is the root and the node with ID 11 seen in Figure 4 and Figure 5, at first, is a common valid node, thus enabling the simulation of the network without the presence of attacks. But to simulate

the presence of the attacks on the network the node with ID 11 has been modified to act as the malicious node.

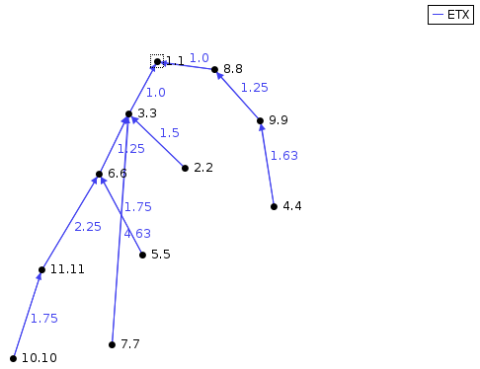


Figure 5. The Directed Acyclic Graph of the simulated network.

A. The simulation without attack

In this simulation, we do not have attacks and all nodes are valid. As can be easily seen in Figure 6, all nodes have the consumed power of less than 10%.

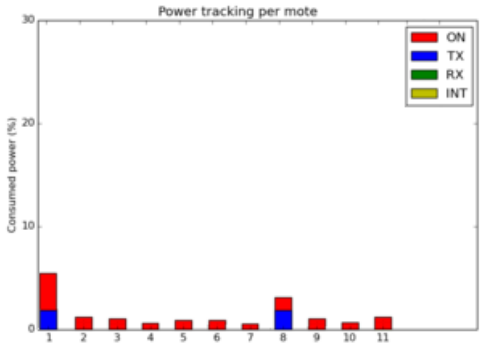


Figure 6. The Consumed Power without attack.

B. The simulation of the Black Hole attack

In this simulation the malicious node dropped all the collected data application messages instead of forwarding them. When using the Selective Forward variant in the simulation, only the received data plane messages of some nodes with IP specified by the attacker node were dropped. This way, it was easily observed a malfunction in the network in relation to packets delivery and packet integrity.

The Black Hole attack can also be enhanced if combined with a sinkhole attack. When simulating the Black Hole with the sinkhole variant, the DAG was changed. Some valid nodes (ID 4, 5 and 10) in the neighborhood of the malicious node (ID 11) have now set it as their parent. This way, the attack has become even more efficient, because it is listening and dropping a larger number of the received messages. The DAG changed can be seen in Figure 7 generated from the Cooja Simulator.

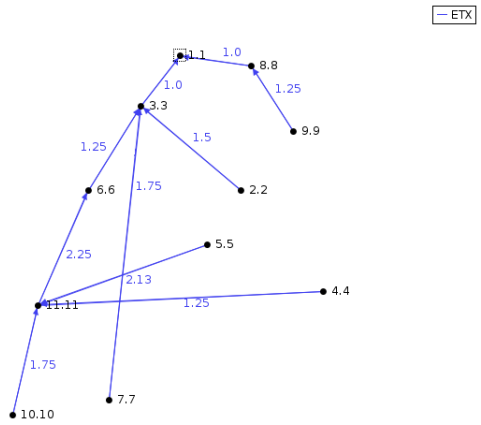


Figure 7. The DAG of the simulated network with Sinkhole attack.

C. The simulation of the Flooding attack

In this simulation the malicious node (Node with ID 11) impacts nodes with IDs 4, 5, 6 and 10 (Figure 4). It is very easy to see in Figure 8 that these nodes are particularly affected by the attack in terms of ON and RX times and the malicious node consumed a lot power with TX. So these nodes spend a lot of energy and memory, to read the requests sent by the malicious node. The power consumed by the attacker node and by the nodes affected by the attack is well over 10% but, the power consumed by other nodes remains below 10%.

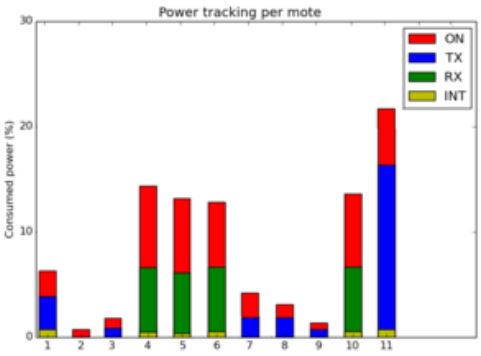


Figure 8. The Consumed Power with Flooding attack.

D. The simulation with our Architecture to isolate the attacker node

During the execution phase, our architecture is intended to isolate the attacker node so that it does not cause further damage to the network. When simulating the isolation of the attacker node, the DAG was changed and another node, besides the malicious node (ID 11) was also isolated.

The other node also isolated can be seen in Figure 9. This other node was the with ID 10. This occurred because the node with ID 10 was very far from the others.

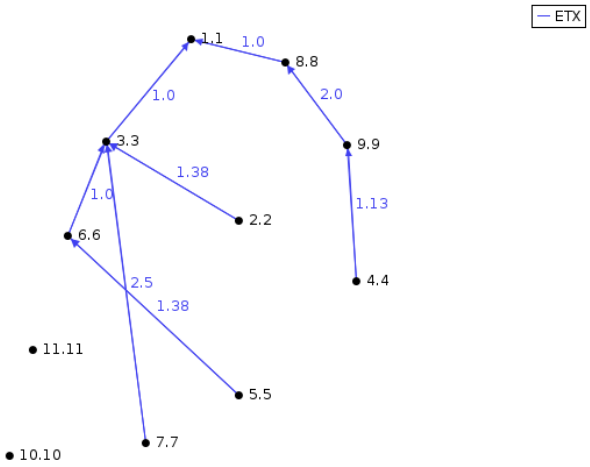


Figure 9. The DAG of the simulated network.

IX. CONCLUSION

In this paper, we have used the MAPE-K control loop to design a new self-protecting architecture for the IoT. With this architecture, it will be possible to incorporate security facilities in the IoT systems releasing the programmer to treat only the functional requirements. Besides, the user can extend the architecture developing new security services to handle specific attacks.

The research will help provide a self-protection mechanism for IoT networks, facilitate the detection and the classification of possible attacks on smart devices, mitigate the damage of the attacks suffered ensuring better performance and increasing the confidence of users when using devices connected to IoT network.

The Self-Protecting architecture deals with five different attacks (Sinkhole, Selective forward, Black Hole, Flooding and Hello Flood) bearing in mind the memory consumption and energy due to a lack of resource of the available devices in the IoT environment.

The performance of the system should be evaluated to verify if the Self-Protecting architecture has better results than related work. New attacks and new technologies will emerge, and then the work presented here may be extended to address those.

ACKNOWLEDGMENT

This work was supported by CAPES and FAPITEC/SE

REFERENCES

[1] D. INFSO, "Internet of things in 2020: Roadmap for the future," INFSO D, vol. 4, 2008.

[2] D. Niyato, E. Hossain, and S. Camorlinga, "Remote patient monitoring service using heterogeneous wireless access networks: architecture and optimization," IEEE Journal on Selected Areas in Communications, vol. 27, no. 4, 2009.

[3] O. Vermesan, P. Friess, and A. Furness, "The internet of things 2012: New horizons," IERC 3rd edition of cluster book, 2012.

[4] J.-P. Vasseur and A. Dunkels, Interconnecting smart objects with ip: The next internet. Morgan Kaufmann, 2010.

[5] M. R. Nami and M. Sharifi, "Autonomic computing: a new approach," in Modelling & Simulation, 2007. AMS'07. First Asia International Conference on. IEEE, 2007, pp. 352–357.

[6] J. Kephart, G. Sorkin, M. Swimmer, and S. White, "Blueprint for a computer immune system," in Artificial immune systems and their applications. Springer, 1999, pp. 242–261.

[7] S. R. White, M. Swimmer, E. J. Pring, W. C. Arnold, D. M. Chess, and J. F. Morar, "Anatomy of a commercial-grade immune system," IBM Research White Paper, 1999.

[8] G. Portokalidis and H. Bos, "Sweetbait: Zero-hour worm detection and containment using low-and high-interaction honeypots," Computer Networks, vol. 51, no. 5, 2007, pp. 1256–1274.

[9] M. Swimmer, "Using the danger model of immune systems for distributed defense in modern data networks," Computer Networks, vol. 51, no. 5, 2007, pp. 1315–1333.

[10] S. Rawat and A. Saxena, "Danger theory based syn flood attack detection in autonomic network," in Proceedings of the 2nd international conference on Security of information and networks. ACM, 2009, pp. 213–218.

[11] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," Ad hoc networks, vol. 11, November 2013, pp. 2661–2674.

[12] D. M. Shila, Y. Cheng, and T. Anjali, "Mitigating selective forwarding attacks with a channel-aware approach in wmns," Wireless Communications, IEEE Transactions on, vol. 9, May 2010, pp. 1661–1675, doi:10.1109/TWC.2010.05.090700.

[13] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," Computer, vol. 36, January 2003, pp. 41–50, doi:10.1109/MC.2003.1160055.

[14] D. Weyns, S. Malek, and J. Andersson, "Forms: a formal reference model for self-adaptation," in Proceedings of the 7th international conference on Autonomic computing. ACM, June 2010, pp. 205–214, doi:10.1145/1809049.1809078.

[15] T. Winter, "Rpl: Ipv6 routing protocol for low-power and lossy networks," 2012.

[16] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer networks, vol. 54, October 2010, pp. 2787–2805, doi:10.1016/j.comnet.2010.05.010.

[17] D. Martins and H. Guyennet, "Wireless sensor network attacks and security mechanisms: A short survey," in Network-Based Information Systems (NBiS), 2010 13th International Conference on. IEEE, November 2010, pp. 313–320, doi:10.1109/NBiS.2010.11.

[18] P. Goyal, S. Batra, and A. Singh, "A literature review of security attack in mobile ad-hoc networks," International Journal of Computer Applications, vol. 9, November 2010, pp. 11–15.

[19] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," Computer, vol. 35, December 2002, pp. 54–62, doi:10.1109/MC.2002.1039518.

[20] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the ip-based internet of things," Wireless Personal Communications, vol. 61, September 2011, pp. 527–542, doi:10.1007/s11277-011-0385-5.

[21] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," Ad hoc networks, vol. 1, September 2003, pp. 293–315.

[22] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the rpl-based internet of things," International Journal of Distributed Sensor Networks, vol. 2013, June 2013.

[23] R. de AC Mello, A. de RL Ribeiro, F. M. de Almeida, and E. D. Moreno, "An architecture for self-protection in internet of things," ICWMC 2016, 2016, p. 51.

[24] F. M. de Almeida, A. d. R. L. Ribeiro, and E. D. Moreno, "An architecture for self-healing in internet of things," UBICOMM 2015, 2015, p. 89.

ICWMC 2016 : The Twelfth International Conference on Wireless and Mobile Communications (includes ICDT, ICSEA, ICMC 2016)

APÊNDICE B – ICWMC-B3

An Architecture for Self-protection in Internet of Things

Ruan de A. C. Mello, Admilson de R. L. Ribeiro, Fernando M. de Almeida, Edward D. Moreno
Department of Computing
Federal University Sergipe UFS
São Cristóvão, Brazil
E-mails: ruanmello@gmail.com, admilson@ufs.br, fernando.m.al.91@gmail.com, edwdavid@gmail.com

***Abstract*—To make the Internet of Things a more receptive environment and well regarded by everyone, it is important to invest in security. The devices involved in the Internet of things environment have limited computational resources and expose the network to many threats. With the use of an architecture to be able to detect, classify and mitigate the effects of these threats, it is possible to create a safer environment. This paper proposes a security architecture for the Internet of Things, considering the limited computational resources of the environment in question. This Architecture uses the Dendritic Cells Algorithm (DCA) combined with a neural network to detect attacks and use the White List to ignore the malicious nodes in the network.**

Keywords–Internet of Things; Autonomic Computing; Self-Protection.

I. INTRODUCTION

Next ages of computing will tend to be beyond the traditional work environment. The Internet of Things (IoT) is a recent paradigm that makes part of this new age and its main goal is to create the possibility of communication between people and things and also between things without the need of human intervention [1].

Atzori et al. [2] calls attention to the impact of the fast advance of IoT and the great challenges that follow such as the interoperability between devices, the limited computational resources and especially the security. According to Roman et al. [3] the Internet and the users are under constant attacks, but there are many business models that try to provide the ethical and safe use of the Internet. The IoT environment is considered more vulnerable than the conventional Internet [2]. This occurs because when the wireless network has many nodes, it becomes easier for both physical and logical attacks, since it is not possible to apply one complex security mechanism due to lack of computational resources. There are many malicious models and many others will emerge associated with this environment. The challenge is to prevent the growth of such models or at least to minimize their impact.

Dobson [4] highlights the relevance of the autonomic characteristics, considering the growing amount of devices interconnected in the IoT environment. In 2011, the number of connected devices already exceeded the real number of people around the world and it is estimated that in 2020 this number will reach 24 billion.

The Vice President - Senior of International Business Machines (IBM), Paul Horn, introduced in March 2001, for the first time the use of the term Autonomic Computing [5]. Horn deliberately chose a term with a biological connotation trying to compare in this manifesto the need of self-management in complex systems aimed to reduce the burden on system administrators with the way that the autonomic nervous system regulates the heart beat and body temperature [5]. Thus, the conscious brain will be released from the burden of dealing with these and many other functions that can be considered low-level, but vital for your functioning. In this manifesto, he presented the four properties of self-management: self-configuring, self-optimizing, self-healing and self-protection.

Our work completes the architecture proposed by Almeida et al. [6] addressing the self-protection on the IoT. The proposed architecture consists of five modules. The tasks are distributed among the modules in order to share the responsibilities. By dividing responsibilities between modules, it makes easier to design self-protection systems for the IoT environment. This Architecture uses the dendritic cell algorithm (DCA) combined with a Multilayer Perceptron Neural Network with Limited Weights (MLPLW) to detect attacks in the network. The MLPLW can learn non-linear patterns during the execution, making the attack detection more efficient. In case of attack on the network of IoT, the architecture proposed will discover the type of attack and minimize the damage.

The remainder of this document has seven sections: Section II presents five common types of attacks to the IoT environment that will come to be mitigated with the proposed architecture; Section III introduces some aspects of autonomic computing and presents the autonomic loop MAPE-K; Section IV presents the Dendritic Cells Algorithm; Section V describes the architecture proposed; Section VI presents two related works to Internet security and the system of self-protection; Section VII presents the initial results; Section VIII concludes the paper.

II. ATTACKS IN INTERNET OF THINGS

According to Atzori [2] the characteristics of IoT (low power, limited energy and limited resources) expose the network to many threats. Most of them attack the limited power of the sensors. In other cases these threats modified or deleted some data. Following this section, we will discuss some of the latest and more common attacks on the environment of the IoT and wireless sensor networks [7].

1) Sinkhole: In a Sinkhole attack, the attacker tries to attract all the traffic from neighboring nodes [8]. So, practically, the attacker node listens to all data transmitted from neighboring nodes. Only this attack does not cause too much damage in the network, but together with another type of attack (Selective Forward or Black Hole), can become very powerful.

 ISBN: 978-1-61208-514-2

2) Selective forward: In a Selective Forward attack, the attacker node receives the transmission packets, but refuses to transmit some of them and drops those that it refused to transmit. The attacker must choose which packets to discard according to some standard such as size, destination or origin [9]. In this case, only the packets released by the attacker node can be freely transmitted.

3) Black Hole: In a Black Hole attack, the attacker node receives the transmission packets and drops all packets received, regardless of type, size, origin or destination [9].

4) Flooding: There are vulnerabilities related to the exhaustion memory. One manner to take advantage of this vulnerability is when an opponent sends too many requests trying to connect to the victim, every request makes the victim allocate the resources in an attempt to maintain the connection [10]. Thus, to prevent the total resource depletion is necessary to limit the number of connections. However, this solution also prevents valid nodes to create a connection with the victim, causing problems such as queuing [10].

5) Hello Flood: The attack Hello Flood uses a device with a powerful signal to regularly send some messages; that way, the network is left in a state of confusion [7]. In order to find ad-hoc networks, many protocols use Hello Messages for discovering neighbor nodes and automatically create a network. With the Hello Flood attack, an attacker can use a device with high transmission power to convince every other node in the network that the attacker is its neighbor, but these nodes are far away from the attacker. In this case the power consumption of sensors is significantly increased, because of protocols that depend on exchange information between neighbor nodes for topology maintenance or flow control [7].

Previously, we saw some of the most common attacks on IoT networks and in the next topics will be analyzed the possible strategies to end or to mitigate the damage caused by them.

To stop the damages caused by attacks on a network, first it is necessary to detect these attacks, using an intrusion detection system (IDS). An IDS analyzes network activity and attempts to detect any unusual behavior that may affect the integrity of the network. Based on information provided by IDS, strategies are created to cope the attacks. For example:

- To mitigate Sinkhole - If the geographical locations of the nodes of RPL DODAG are known, the effect of Sinkhole attacks can be mitigated by the use of flow control, making sure, that the messages are traveling to the correct destination. The RPL protocol also supports multiple instances DODAG offering alternative routes to the root DODAG [11].
- To mitigate Hello Flood - A simple solution to this attack, it is perform a bidirectional check for each message ”HELLO” [12]. If there is no recognition, the path is assumed to be bad and a different route is chosen. If geographical locations of the nodes of RPL DODAG are known, all packets received from a node that is far beyond of the common network node transmission capacity can be dropped.
- To mitigate Selective Forward - An effective countermeasure against Selective Forward attacks is to ensure that the attacker cannot distinguish the different type of packets, forcing the attacker to send all or none packets [13].

Raza et al. [14] said that the most efficient and fastest way to stop the damage of routing attacks is isolate the malicious node. Some forms to ignore the attacker node were studied. These forms are:

- The Black List: After identifying the nodes and find the attackers, it will be created a list and all the malicious nodes will be added in order to exclude them from the possible routes of traffic data. To ignore the attacker, it will be done a verification in the Black List excluding all nodes found of the typical RPL DODAG that have a root and multiple nodes.
- The Gray List: After identifying the nodes and find the attacker, it will be created a list. The suspicious attacker node will be added to this list with the intention of excluding it from the possible routes of traffic data, for a predetermined time. After the end of the predetermined time the suspicious attacker node is deleted from the list. In this way, if have any doubt about the identification of the attacker node, the node may re-join the network. To ignore the suspicious attacker nodes, when create the routing, it will be done a verification in the Gray List excluding all nodes found of the typical RPL DODAG that have a root and multiple nodes.
- The White List: As in the example of the Black List, it will be created a list after identifying the nodes and find the attacker node. But this time will be added into the White List only the valid nodes and all malicious nodes will be excluded. This way will have a verification stating which nodes are valid and must belong to a typical RPL DODAG with a root and several nodes.

III. AUTONOMIC COMPUTING

The initiative of autonomic computing proposed by IBM, is based on the human nervous system [4]. The idea is, as the body’s mechanisms that have functions with self-management and do not demand any conscious act such as heartbeat or intestinal activity, a computer system creates mechanisms that will also allow it to have a self-management [4].

According to the manifest proposed by IBM in 2001, there are four self-managing properties: self-configuration, self-healing, self-optimizing and self-protection [5].

- Self-Configuration Through self-configuring autonomic system will be installed and set up to attend the high level predetermined policies according to user intentions.
- Self-Healing Through this property it is possible for autonomic systems detect, diagnose and repair local problems resulting from bugs or failures in software and hardware.
- Self-Optimization The property of self-optimization is present on systems that can perform some change, proactively, to improve the performance and quality of service.

 ISBN: 978-1-61208-514-2

- Self-Protection The self-protection property is present in the systems that can defend themselves from malicious attacks and unauthorized changes. The autonomic system with self-protection is used to prevent and anticipate security breaches.

Dobson [4] say that self-management mechanisms in the autonomic computing are not independent entities. For example, the success of an attack against the system, requires actions of self-healing, self-configuration and self-optimization initially to ensure, that the system will have a trusted operation. After that, the self-protection would have responsibility for dealing with similar attacks in the future.

A. MAPE-K Autonomic Loop

The MAPE-K loop was presented by IBM as a reference model. Composed of five modules that can be seen in Fig. 1, the MAPEK-K Loop is intended to distribute the tasks of each element of the autonomic computing [5]. The modules that build the MAPE-K Loop are, respectively, monitoring, analysis, planning, execution and knowledge.

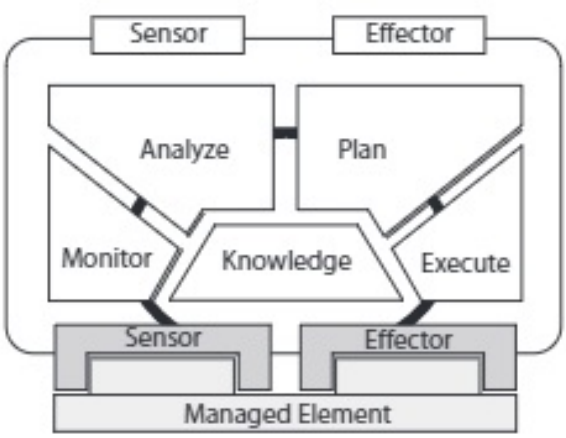


Figure 1. Mape-K Autonomic Control Loop [15].

- The Monitoring module uses sensors to collect data from the managed element, which could be a software or hardware resource, or an autonomic manager itself.
- The Analysis module provides mechanisms to interpret the collected data from the monitoring phase and predict future situations.
- The Planning module builds the necessary actions to achieve the goals.
- The Execution module uses effects to make changes on managed elements.
- The Knowledge module is in charge of keeping relevant data in the memory to accelerate decision making.

IV. DENDRITIC CELL ALGORITHM

The Dendritic Cell Algorithm (DCA) was introduced by Greensmith [16] and is inspired by the Danger Theory regarding the human Immune System. The main elements of DCA are Dendritic Cells (DC), the lymph node and antigens. The DC input signals are signs of danger, safe signals, PAMP signals, signs of inflammation. The DC output signals are: migration signal (costimulatory Molecules-CSM), semi-mature signal and mature signal.

Iteratively, the antigens are presented to DC. All signals increment the migration signal indicating when the dendritic cells will migrate to the lymph node and being processed. The danger signs and PAMP increment the mature signal of the DC while the safe signal increments the semi-mature signal of the DC. The sign of inflammation potentiates the growth of all signals.

When DC reaches the migration threshold is sent to the lymph node, and DC will be labeled how mature if the mature signal is greater than semi-mature or semi-mature otherwise. When the lymph node has a number of DCs, the antigens anomaly index is calculated, the Mature Context Antigen Value (MCAV) from the equation (1) where M is the number of mature DCs and SM the number of semi- mature DCs.

$$MCAV = \frac{M}{(SM + M)} \quad (1)$$

If MCAV has a value greater than the threshold, the DCA detects the presence of an intruder.

V. SELF-PROTECTION ARCHITECTURE

The Self-protection architecture for the IoT proposed in this work has five modules (Monitoring, Analysis, Planning, Executing and Knowledge) and was based on the MAPE-K loop. It is important to remember that the main contribution of this work is the implementation of the missing modules (Planning and Execution). This way, we will be completing the architecture proposed by Almeida et al. [6].

The monitoring and analysis modules are responsible, respectively, for collecting through the sensors, some information of the network that will be analyzed to measure the possibility of being associated with an attack. These two modules are present in the network nodes and in the border router (6BR). The planning and execution modules will be responsible, respectively, for identifying the attacker, the type of attack and to mitigate the damage in the network. The information listed as relevant data for analysis, planning and execution is: type of transport protocol, type of application protocol, time of communication, number of messages sent, number of messages effectively sent and number of messages received.

In Fig. 2, we can see that the border node (6BR) will have all components. This occurs because the 6BR has sensing and also for being the main element of the network. The components of the monitoring module and a part of the analysis module components are present in the network nodes that have the self-protection system. The components of the planning and execution modules are present only in 6BR, but the need to distribute them among the other network nodes can be considered. The five modules present in the architecture will be described more clearly in the following subsections.

1) Monitoring Phase: The components responsible for the monitoring phase are present in all network nodes, especially in 6BR. At this stage, the network information and the nodes are monitored. Some important information collected during the sensing to future analysis is: number of successfully sent

 ISBN: 978-1-61208-514-2

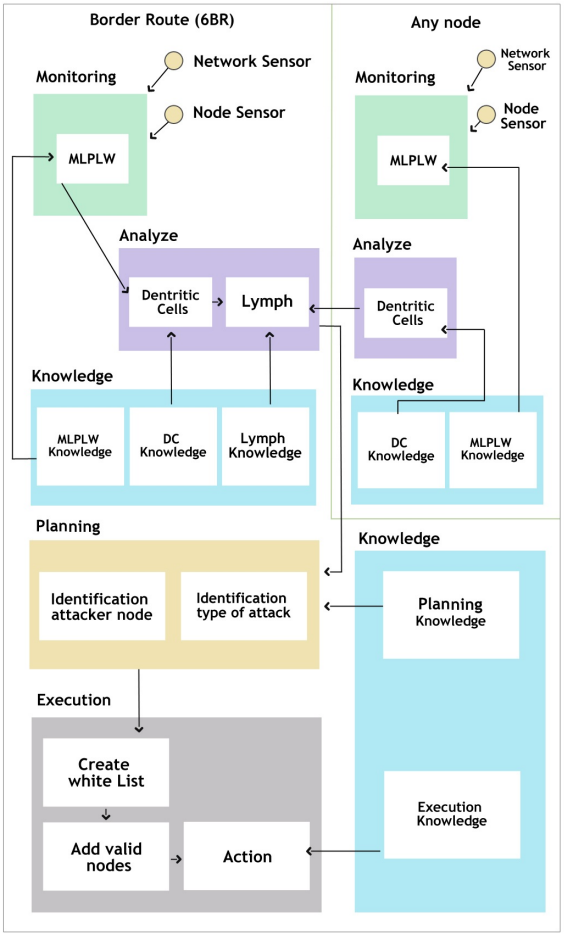


Figure 2. Self-Protection Architecture.

packets, number of sent packets, number of lost packets and the information about routing in DODAG tree. The main component of the monitoring phase, in this architecture proposed, is the Multiplier Perceptron Neural Network with Limited Weights (MLPLW), that can learn non-linear patterns during the execution. The MLPLW in this architecture is used to predict what information received indicates a problem in the node or in the network, for example, the MLPLW can predict the decrease of successfully sent packets from the packet sender address, if the training is enough. The prediction of the MLPLW is sent to the Analysis phase, to the Dendritic Cell Algorithm.

2) Analysis Phase: The component responsible for analysis phase uses the Dendritic Cell Algorithm (DCA), which is inspired by the danger theory of the human immune system. This algorithm has a high distributive potential and a low false positive rate. The DCA is used to analyze the results received from the MLPLW and determine if an attack is happening. The Dendritic Cells collect information from the MLPLW and migrates to the Lymph when the cell becomes mature or semi-mature. The Lymph receives mature and semi-mature dendritic cells and determine if the system is in danger. This information indicates an attack or not and, if there is an attack, it is sent to the Planning phase.

3) Planning Phase: The component responsible for the planning phase needs confirmation that really exist the presence of attack on the network, otherwise it becomes obsolete. Because the main objective is to find out what type of attack occurred on the network and to identify the attacker node through the information acquired by the analysis module. Thus, the execution module will decide the most appropriate action to be taken to mitigate the damage caused by the attack. The type of attack will be classified according to the following set of options: route selection group, packets control group and exhaustion attack group.

Through the obtained results, it is possible to differentiate the type of attack occurred. For example:

- If an attack belongs to the route selection group (Sinkhole), it will be taken into account the flow of messages in the legitimate nodes and in the attackers nodes. In this case, it will be needed to evaluate the influence of attackers in the choice of a better route. For this, it is necessary to compare the flow of messages from legitimate nodes regarding to the attackers nodes, through a medium and total number of messages.
- If an attack that belongs to the packet control group, which includes messages disposal (Black Hole and Selective Forward), it will be taken into account the number of messages sent, messages effectively sent and messages received.
- If an attack that belongs to an exhaustion attack group (Hello Flood and Flooding), it will be taken into account the number of packets sent by the attacker node, which is intended to send a huge number of messages to overload the system.

4) Execution Phase: The component responsible for the execution phase should mitigate or stop the damage caused by the attacks occurred on the network. The type of attack and the identification of the attacker node will be information that will influence in the choice of the predetermined action to mitigate or stop the damage in the network. These two important information will be provided by the Planning Phase. The reason to find out the attacker node is, trying to isolate as quickly as possible and create a new route, thus, avoid further damage to the network. The type of attack will be among one of the three groups mentioned above. According to the group selected there will be a specific action to solve the problem, because at each type of attack causes different types of damage on the network.

The first action to be taken by the components of the execution module, since the attack was detected, it is to ignore the malicious node. To perform this action it is important to be made the identification of network nodes, legitimate and malicious. Raza et al. [14] say that it is necessary to be careful with the way of identifying nodes. If possible it should be avoided the identification by IP address or MAC address, because they can be easily falsified. After making the identification of the nodes, some ways to ignore the attacker

 ISBN: 978-1-61208-514-2

node were studied. Three ways to isolate the malicious node were analyzed before choosing the most convenient. The three ways are: Black List, Gray List and White List.

The way chosen was the White List, because the maintenance of this list is simple. This way, all valid nodes will recalculate their rank in the RPL protocol (DODAG). To recalculate the rank of all valid nodes, it will be necessary to ignore the DODAG Information Object (DIO) of all nodes with higher rank than theirs and of nodes that are not present in the White List. Thus, for a stranger node join in the network, it should be reported as safe and added in the White List.

5) Knowledge Phase: The components of knowledge phase will be responsible for keeping all the knowledge acquired by the system. Knowledge about the planning and execution modules will be at 6BR. The information kept by the Knowledge Module will be used to facilitate and accelerate the discover of the type of attack, the attacker node and the action to be taken to protect the network.

VI. RELATED WORK

The related works listed in this paper not only detect the attack, but also tries to mitigate the damage caused by the attacker. The related works will be described more clearly in the following subsections.

A. CAD

In [9], the authors describe how they developed a way to detect, identify the attacker node and classify the attack in a Wireless Mesh Networks (Wireless Mesh Networks - WMNs). They considered a special case of denial of service (DoS), known as Selective Forward, where the attacker node behaving maliciously pushing forward only a subset of the packets that received but discards the others.

While most studies about Selective Forward focuses on attack detection assuming a wireless channel, error free, the authors Shila et al. [9] considered more interesting the challenging scenario where the fall of the package may be due to an attack or normal loss events, such as the collision of access to the medium or bad quality of the channel.

Specifically, they developed an algorithm that can effectively distinguish the Selective Forward from the packet losses occurring in normal loss events. The developed algorithm is called CAD (Channel Aware Detection) and is based on two strategies, channel estimation and traffic monitoring. If the loss rate monitored in certain jumps exceeds the normal rate of estimated loss, the nodes involved will be identified as attackers. In addition, they perform analytical studies to determine the optimal detection thresholds that minimize the sum of false alarm and missed detection probabilities.

The traffic control procedure works basically that way, each intermediate node, monitors along the path the behavior of the neighbors. Given a path determined by the routing protocol, the CAD sends messages along the way to probe, to detect possible attacks. In the event of a positive detection, CAD, then triggers the underlying routing protocol to activate a process for finding out a new route.

CAD design was used by Network Simulator NS2 Berkeley (v2.29) for simulations. By the end of the simulation, using the CAD algorithm, the results showed that, in the presence of normal loss events or without, CAD can detect and classify the attack, increase the packet delivery rate in the network and finding out attacker nodes.

B. SVELT

SVELT, name given by authors, Raza et al. [14] to Intrusion Detector System for the Internet of Things projected by them, is applied in 6LoWPAN networks that use the RPL routing protocol. The approach of SVELTE is distributed and centralized, inserting modules in the edge router (6BR) and network nodes. The three main modules SVELTE are 6LoWPAN Mapper, Component Detector Intruder and distributed mini - firewall. The Mapper 6LoWPAN (6Mapper) reconstructs the routing tree RPL protocol (DODAG) in 6BR, each network node has a client for the 6Mapper. To rebuild the tree DODAG, you must periodically send requests to the 6Mapper customers, who respond to their corresponding information: NodeID, ParentID, Neighbor IDs and ranks of Neighbors IDs. It should be considered whether the network uses some form of reliable communication authentication, so you can reduce the size of requests and responses.

Once the SVELT detects an attack, the goal is to mitigate its effects and remove the intruder from the network, Raza et al. [14] say that the simplest approach to remove an attacker is to ignore it. Taking this approach requires the identification of the attacker node. The authors adopted, as a way to ignore the malicious node the White List, which would include all valid nodes and would reject malicious nodes. This method is reliable and easy to maintain in the presence of many attackers. As a result, in SVELTE a white list is used. The informed location of the nodes will also help mitigate the SVELTE Sybil and CloneID attacks intended to disrupt the routing information forging identities.

Implementations of SVELTE and mini-firewall were made in ContikiOS, the code is open and available. Raza et al. [14] used the ContikiOS and its RPL implementations of 6LoWPAN and IP stack. The evaluation was made empirically using Cooja, the network simulator of the ContikiOS, measuring the rate of detection, true positives and false positives for each experiment.

The authors concluded that the SVELTE is very effective for Sinkhole and Selective Forward attacks on a network with fewer losses. When the network has more losses, the system has better results when the RPL network becomes stable. Getting rates close to 90% on a network with losses when the RPL network stabilizes and results close to 100% on a network without loss. As for energy consumption, SVELTE solution consumes 30% more energy than using only the RPL. As the consumption of memory, 6Mapper client has 1414 bytes of overhead, the Firewall client has 246 bytes, 6Mapper server varies with the number of nodes and neighbors: 3580 bytes for one node and one neighboring, 3846 bytes into 8 nodes and 1 neighbor, 4152 bytes into 16 nodes and 1 neighbor and 4724 bytes into 16 nodes and 8 neighbors.

C. Comparison of Related Work

The SVELTE, as the authors claim, is the first IDS (Intrusion Detector System) for the IoT. The work presented has a huge contribution to design an IDS with the characteristics of a network for IoT, considering the technologies used in the communications stack, such as 6LoWPAN and RPL routing

 ISBN: 978-1-61208-514-2

protocol. However its approach does not have autonomic characteristics for auto protect the network from further attacks, only the determined attack on the network project level.

TABLE I. COMPARISON OF RELATED WORK.

Qualities	*SVELT*	*CAD*	*Architecture proposal*
Designed for IoT	X	-	X
Autonomicity	-	X	X
Mitigate SinkHole	X	X	X
Mitigate Selective Foward	X	X	X
Mitigate BlackHole	X	X	X
Mitigate Flooding	-	-	X
Mitigate Hello Flood	-	-	X

Like the SVELT, CAD not only detects the attack, but also tries to mitigate the damage caused by the attacker. The CAD is directed to the WMNs and can differentiate between losses occurring in the normal events of a legitimate attack Selective Forward efficiently. In Table 1, we present the differences between the related works and the proposed architecture.

VII. RESULTS

The initial results of this research are about the technique used in the monitoring phase. The chosen technique for the first efforts is an ANN (Artificial Neural Network), a MLPLW based on the neural network with limited precision weights [17].

The MLPLW implemented has 10 neurons in the hidden layer and each weight is represented by a byte. The training technique of the MLPLW is the QBPSS (Quantized Back-Propagation Step-by-Step) [17], a modified version of Back-Propagation for neural network with limited weights.

TABLE II. MEMORY CONSUMPTION.

Resource	*MLP*	*MLPLW*	*MLPLW with training*
ROM Memory	214 bytes	354 bytes	1716 bytes
RAM Memory	3360 bytes	420 bytes	420 bytes

The KDD99 dataset it is widely used in Intrusion Detection Systems. Thus, KDD99 dataset was used with our MLPLW ANN implementation with a stream based training [18]. Each input is used once and the accuracy and false positive rates were measured every thousand inputs. After the first thousand inputs, the MLPLW achieved 97,65% accuracy, but oscillated until the thirty-fourth thousand input. The oscillation of the accuracy rate of the MLPLW is depicted in Fig. 3.

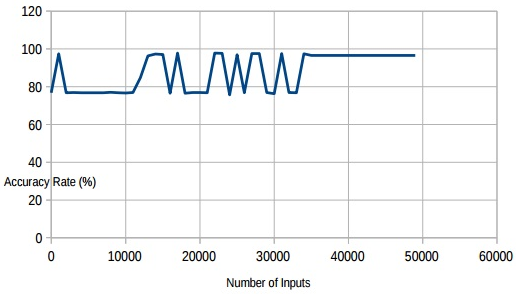


Figure 3. Accuracy rate of MLPLW.

As our proposed architecture shall be used in the Internet of Things context, the used techniques should be in accordance to the use of resources. It is possible to see in Table II, that the MLPLW have a small impact in memory utilization in an ARM Cortex-M3 with 512 KB of Persistent memory and 64 KB of Volatile memory.

VIII. CONCLUSION

The completion of this proposal will help provide a self-protection mechanism for IoT networks, facilitate detection and the classification of possible attacks on smart devices, mitigate the damage of the attacks suffered ensuring better performance and increase the confidence of users when using devices connected to IoT network.

The architecture proposed deals with five different attacks (Sinkhole, Selective forward, Black Hole, Flooding and Hello Flood) bearing in mind the memory consumption and energy due to a lack of resource of the available devices in the IoT environment. This Architecture uses the Dendritic Cell Algorithm combined with a neural network (MLPLW) to detect attacks and use the White List to ignore the malicious nodes in the network.

The MLPLW implemented shows that the monitor phase of the proposed architecture can use less than 1% of the memory of the embedded system and still have a high accuracy rate.

For future work, there will be the implementation of the proposed architecture to validate it. The performance of the system should be evaluated to verify if the proposed architecture has better results than related work. New attacks and new technologies will emerge, and then the work in question may be extended becoming increasingly complete.

ACKNOWLEDGMENT

This work was supported by CAPES and FAPITEC/SE

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (iot): A vision, architectural elements, and future directions,” Future Generation Computer Systems, vol. 29, September 2013, pp. 1645–1660, doi:10.1016/j.future.2013.01.010.

[2] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” Computer networks, vol. 54, October 2010, pp. 2787–2805, doi:10.1016/j.comnet.2010.05.010.

[3] R. Roman, P. Najera, and J. Lopez, “Securing the internet of things,” Computer, vol. 44, September 2011, pp. 51–58, doi:10.1109/MC.2011.291.

[4] S. Dobson, R. Sterritt, P. Nixon, and M. Hinchey, “Fulfilling the vision of autonomic computing,” IEEE Computer, January 2010, doi:10.1109/MC.2010.14.

[5] J. O. Kephart and D. M. Chess, “The vision of autonomic computing,” Computer, vol. 36, January 2003, pp. 41–50, doi:10.1109/MC.2003.1160055.

[6] F. M. de Almeida, A. d. R. L. Ribeiro, and E. D. Moreno, “An architecture for self-healing in internet of things,” UBICOMM 2015, July 2015, p. 89.

[7] D. Martins and H. Guyennet, “Wireless sensor network attacks and security mechanisms: A short survey,” in Network-Based Information Systems (NBiS), 2010 13th International Conference on. IEEE, November 2010, pp. 313–320, doi:10.1109/NBiS.2010.11.

[8] P. Goyal, S. Batra, and A. Singh, “A literature review of security attack in mobile ad-hoc networks,” International Journal of Computer Applications, vol. 9, November 2010, pp. 11–15.

 ISBN: 978-1-61208-514-2

[9] D. M. Shila, Y. Cheng, and T. Anjali, "Mitigating selective forwarding attacks with a channel-aware approach in wmns," Wireless Communications, IEEE Transactions on, vol. 9, May 2010, pp. 1661–1675, doi:10.1109/TWC.2010.05.090700.

[10] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," Computer, vol. 35, December 2002, pp. 54–62, doi:10.1109/MC.2002.1039518.

[11] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the ip-based internet of things," Wireless Personal Communications, vol. 61, September 2011, pp. 527–542, doi:10.1007/s11277-011-0385-5.

[12] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," Ad hoc networks, vol. 1, September 2003, pp. 293–315.

[13] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the rpl-based internet of things," International Journal of Distributed Sensor Networks, vol. 2013, June 2013.

[14] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," Ad hoc networks, vol. 11, November 2013, pp. 2661–2674.

[15] D. Weyns, S. Malek, and J. Andersson, "Forms: a formal reference model for self-adaptation," in Proceedings of the 7th international conference on Autonomic computing. ACM, June 2010, pp. 205–214, doi:10.1145/1809049.1809078.

[16] J. Greensmith, U. Aickelin, and S. Cayzer, "Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection," in Artificial Immune Systems. Springer, August 2005, pp. 153–167, doi:10.1007/11536444_12.

[17] J. Bao, Y. Chen, and J. Yu, "An optimized discrete neural network in embedded systems for road recognition," Engineering Applications of Artificial Intelligence, vol. 25, June 2012, pp. 775–782.

[18] A. Carvalho, K. FACELI, A. LORENA, and J. GAMA, "Inteligência artificial–uma abordagem de aprendizado de máquina," Rio de Janeiro: LTC, 2011.

 ISBN: 978-1-61208-514-2

APÊNDICE C – Mapeamento

Conquistando um ambiente seguro em Redes IoT: Um mapeamento sistemático da literatura sobre segurança em redes da Internet das Coisas.

Ruan Mello, Maria Augusta Nunes, Admilson Ribeiro
Departamento de Ciência da Computação (DCOMP)
Universidade Federal de Sergipe UFS
São Cristovão, Brasil
E-mails: ruanmello@gmail.com, gutanunes@gmail.com, admilson@ufs.br

Resumo

Para tornar o ambiente da Internet das Coisas mais receptivo e bem visto por todos é importante investir em segurança. A maioria dos trabalhos relacionados à área de segurança não estão direcionados ao ambiente da Internet das coisas, para poder abrangê-lo é bastante interessante associar mecanismos de segurança com propriedades autonômicas, considerando o crescimento exponencial de dispositivos conectados. Este trabalho mapeia uma grande quantidade de artigos relacionados à segurança no ambiente da Internet das Coisas. Seleciona os considerados mais importantes de acordo com um critério de seleção e exclusão, extraindo resultados para o mapeamento sistemático. A partir desses resultados será possível decidir quais estratégias de segurança devem ser utilizadas em um ambiente que não possui muito recurso computacional e contem muitos dispositivos conectados

Palavras-chave – Internet of things, security, autonomic, protection.

1. INTRODUÇÃO

A nova era da computação tende a ser fora do domínio do ambiente de trabalho tradicional (GUBBI, 2013). A Internet das Coisas (*IoT*) paradigma recente que faz parte dessa nova era tem como principal objetivo segundo os autores Gubbi, Buyya, Marusic e Palaniswami (2013) fazer com que as pessoas se comuniquem com as coisas e que as coisas também criem comunicações entre si sem a necessidade de intervenção humana.

Atzori, Iera e Morabito (2010) chamam atenção para o impacto causado pelo acelerado avanço da Internet das Coisas e os grandes desafios que o acompanham como, por exemplo, a interoperabilidade dos dispositivos, o escasso recurso computacional e sobretudo a segurança. De acordo com os autores Roman, Najera e Lopez (2011) a Internet e seus usuários já estão sob ataque constante, e uma crescente economia repleta de modelos de negócios baseia-se em prover o uso ético e seguro da Internet focando na exploração de fraquezas fundamentais da versão atual.

O ambiente da Internet das Coisas é considerado ainda mais vulnerável do que o ambiente da Internet convencional segundo Atzori, Iera e Morabito (2010) já que incorpora muitos dispositivos físicos que devem ser restritos. Por se tratar na maioria das vezes de uma rede sem fio com um exagerado número de nós, facilita tanto o ataque físico aos dispositivos quanto o ataque lógico já que não é possível implementar um esquema de segurança complexo devido a falta de recurso computacional. De fato, é previsto que surgirão modelos maliciosos engenhosos voltados a esse ambiente, o grande desafio é impedir o crescimento de tais modelos ou pelo menos mitigar o seu impacto.

Dobson (2010) destaca a importância das características autonômicas, levando em consideração a crescente quantidade de dispositivos interligados no ambiente da Internet das coisas. Em 2011 a quantidade de dispositivos conectados já ultrapassava o real número de pessoas em todo o planeta e estima-se que em 2020 esse número deve chegar a 24 bilhões (GUBBI, 2013). Esses valores exorbitantes inviabilizam a gerência humana de segurança para os dispositivos, criando então a necessidade de automatizar essa função.

O vice-presidente sênior da IBM (*International Business Machines*), Paul Horn, introduziu em março de 2001, pela primeira vez a utilização do termo Computação Autonômica. Horn escolheu deliberadamente um termo com uma conotação biológica buscando comparar em seu manifesto a necessidade de autogerenciamento em sistemas complexos que visam diminuir a carga dos administradores do sistema com a forma que o sistema nervoso autônomo regula a frequência cardíaca e a temperatura corporal. Dessa forma estaria libertando o cérebro consciente do fardo de lidar com estas e muitas outras funções que podem ser consideradas de baixo nível, mas de vital importância para funcionamento do mesmo (KEPHART; CHESS, 2003). Nesse manifesto apresentou as quatro propriedades da auto-gerência: auto-configuração, auto-otimização, auto-cura e auto-proteção.

Portanto, esse mapeamento sistemático tem como objetivo identificar e sistematizar as principais técnicas e métodos utilizados para que as redes *IoT* tornem-se mais seguras e necessitem o mínimo possível da intervenção humana. Além disso, será analisado o crescimento da quantidade de publicações relacionadas à área em questão. Para isso, foram mapeados os artigos em bases de dados consideradas importantes na área da computação.

Assim, esse mapeamento está organizado da seguinte forma: a Seção 2 é apresentada a Metodologia adotada nesse mapeamento; Na Seção 3, é apresentada a Análise dos Resultados; Na Seção 4, é apresentada a Conclusão seguida pelos Agradecimentos e Referências.

2. METODOLOGIA

Com propósito de analisar e mapear o estado da arte acerca das técnicas e métodos utilizados para que as redes *IoT* tornem-se mais seguras e menos dependentes da intervenção humana, optou-se para este artigo a metodologia de mapeamento sistemático (*SLM, Systematic Literature Mapping*). De acordo com os autores Petersen, Feldt, Mujtaba e Mattsson (2008), mapeamento sistemático é um estudo que fornece uma estrutura contendo relatórios de pesquisa e resultados que têm sido publicados de forma categorizada, para criar essa estrutura é preciso primeiro executar os seguintes procedimentos, elaborar questões de pesquisa, efetuar busca pelo material desejado e criar métodos para selecionar o conjunto de material mais relevante, só após executar esses procedimentos será possível extrair os resultados e mapeá-los.

Ao seguir os passos descritos por Petersen *et al.* (2008), com o intuito de realizar o mapeamento sistemático, foi necessário descrever a forma como foram definidos os parâmetros para o processo de mapeamento. Assim as subsecções abaixo descrevem de forma detalhada as questões de pesquisa elaboradas, a estratégia de busca utilizada e os critérios de inclusão e exclusão aplicados para filtrar os estudos primários.

2.1 Questões de pesquisa

Pretendendo alcançar o objetivo deste trabalho foram elaboradas as seguintes questões de pesquisa:
Q1) Qual o ritmo de crescimento das publicações relacionadas à segurança na área da Internet das Coisas?

Q2) Quais as estratégias utilizados para que as redes *IoT* tornem-se mais seguras e necessitem o mínimo possível da intervenção humana?

2.3 Estratégias de Busca e de Seleção

Para realizar a busca dos estudos primários, foram identificadas as bases de dados consideradas importantes da área de computação: *IEEE Xplore (IEEE)*, *Science Direct (SD)* e a Biblioteca Digital Brasileira de Computação (BDBComp).

Durante a execução da busca foram utilizadas as ferramentas de filtragem de cada base visando considerar na busca somente o título, resumo e palavras-chave dos artigos, reduzindo assim o número de artigos fora do escopo do ambiente seguro da Internet das Coisas. Entretanto, na base BDBComp foi realizada uma pesquisa simples, uma vez que não havia mecanismo de busca avançado disponível nessa base. Além disso, foram definidas as seguintes palavras chave para busca:

- Em inglês: Inicialmente foram adotadas quatro palavras-chave: "*IoT*", "*security*", "*attack*" e "*autonomic*", porém, durante as pesquisas, percebeu-se que palavras-chave derivadas: "*protection*" e "*threats*", eram frequentes nos artigos analisados, por isso, as mesmas também foram inseridas ao termo de busca. Além disso, percebeu-se que o uso da palavra-chave "*Internet of Things*" apresenta um maior número de artigos relacionados às questões de pesquisa que a palavra-chave "*IoT*". Assim, optou-se por utilizar "*Internet of Things*" no termo de busca.
- Em português: Inicialmente foram adotadas quatro palavras-chave: "*Idc*", "segurança", "ataque" e "autonomicidade", porém, durante as pesquisas, foi percebido que palavras-chave derivadas: "proteção" e "ameaças", eram frequentes nos artigos analisados, sendo assim, as mesmas também foram inseridas ao termo de busca.

Com isso, os termos de busca ficaram definidos por:

- Em inglês: (("*Internet of thing**" *OR* "*IoT*") *AND* ("*Security*" *OR* "*Protection*") *AND* ("*Autonomic*") *AND* ("*attack**" *OR* "*threat**")).
- Em português: (("Internet das coisas" *OR* "Idc") *AND* ("Segurança" *OR* "Proteção") *AND* ("Autonomicidade") *AND* ("ataque" *OR* "ameaças")).

Tabela I. Resultados das buscas nas bases de dados utilizando o termo de busca

Bases de Dados	**Pesquisa em Inglês**	**Pesquisa em Português**
IEEE	47	--
Science Direct	205	--
BDBcomp	--	6
Total	**258**	

2.4 Critérios de Seleção:

Com a finalidade de filtrar artigos relevantes para o objetivo desse mapeamento sistemático foram definidos critérios para inclusão e critérios para exclusão desses artigos. O estudo contou com os seguintes critérios de inclusão:

1) Serão incluídos os artigos que tem por objetivo tornar o ambiente da Internet das Coisas mais seguro;
2) Serão incluídos os artigos que apresentem técnicas utilizadas para detectar ou mitigar as possíveis ameaças e fraquezas da Internet das Coisas.

A confirmação dos critérios de inclusão foi dada a partir da análise do resumo de cada um dos artigos encontrados.

Em paralelo foi realizada a análise dos artigos quanto aos critérios de exclusão, também foram aplicados aos mesmos os critérios de exclusão explicitados abaixo:

1) Serão excluídos artigos que possuam duplicidade;
2) Serão excluídos artigos que não estejam acessíveis.
3) Foram desconsiderados artigos que não compreendiam o escopo do mapeamento sistemático.

Findada a aplicação dos critérios de inclusão e de exclusão dos artigos encontrados, os mesmos foram avaliados. Dos 258 artigos encontrados, 34 foram selecionados para compor os estudos primários, cabe apontar ainda que nenhum artigo duplicado foi encontrado.

Na Tabela II são apresentados os resultados das buscas nas bases de dados utilizando o termo de busca e aplicação dos critérios de seleção. Também pode ser notado que, dos 258 artigos encontrados no processo de busca, após a leitura e aplicação dos critérios de seleção houve um alto índice de redução no número total de artigos, já que apenas 34 deles foram qualificados pelos critérios de seleção. A identificação completa dos estudos primários pode ser encontrada na seção de Referências deste artigo.

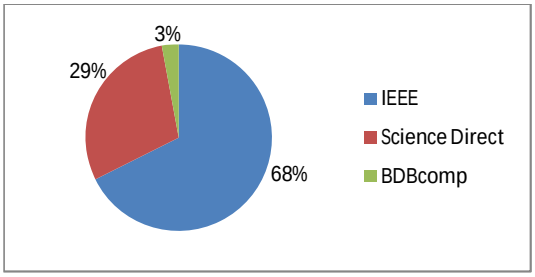
Tabela II. Resultados das buscas nas bases de dados e resultados após aplicação do critério de seleção

Bases de Dados	**Pesquisa**	**Trabalhos Filtrados**
IEEE	47	23
Science Direct	205	10
BDBcomp	6	1
Total	**258**	**34**

A Figura I apresenta o resumo da contribuição de cada base para o total de 34 estudos primários selecionados. Apesar da base *Science Direct* apresentar uma quantidade de artigos bastante superior ao das outras bases, após aplicar os critérios de inclusão e exclusão houve um alto índice de redução nessa quantidade, fazendo com que essa base represente 29% dos estudos primários. A base *IEEE* teve um bom

aproveitamento após a aplicação dos critérios de seleção representando 68% dos estudos primários selecionados e a base BDBcomp compõe apenas 3%.

Figura 1. Estudos primários selecionados de cada uma das Base de Dados



3. ANÁLISE DOS RESULTADOS

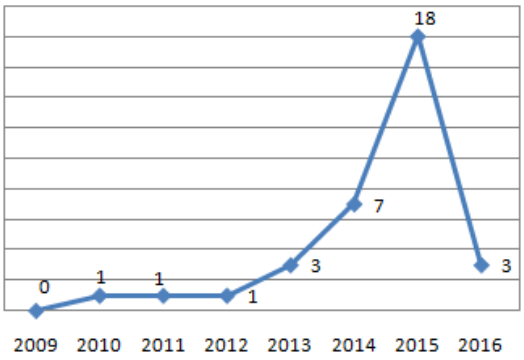
Nesta seção, são apresentados os resultados da análise dos estudos primários, respondendo assim às questões de pesquisa delineadas na seção II deste mapeamento.

Q1) Qual o ritmo de crescimento das publicações relacionadas à segurança na área da Internet das Coisas?

Essa questão visa apresentar o ritmo de crescimento das publicações relacionadas à segurança no ambiente da Internet das Coisas. Dessa forma, é possível tornar o ambiente em questão mais receptivo e bem visto por todos, atraindo mais pesquisadores a investirem tempo e recurso nessa área.

Na Figura 2, é exibido um gráfico que apresenta o grande aumento no ritmo de publicações ao passar dos anos. É muito perceptível o crescimento no número de publicações, visto que, entre 2009 e 2012 foram publicados 3 artigos e entre 2013 e 2016 foram publicados 31 artigos. Por se tratar de um paradigma considerado recente o gráfico iniciou no ano de 2009.

Figura 2. Publicações por ano (2009 a 2016)



Os resultados apresentados nessa seção foram inferidos a partir da leitura aprofundada e análise dos estudos primários selecionados. Em virtude disso, artigos publicados que não passaram pelos critérios de inclusão e ou se encaixaram nos critérios de exclusão não foram contabilizados. É importante destacar que os artigos que compõem os estudos primários selecionados foram publicados nas bases até junho de 2016.

Q2) Quais as estratégias utilizados para que as redes IoT tornem-se mais seguras e necessitem o mínimo possível da intervenção humana?

Essa questão de pesquisa visa projetar as principais estratégias utilizadas para criar técnicas e métodos que tenham o objetivo de tornar o ambiente da internet das coisas mais seguro, abordando análises, modelos de segurança e sistemas detectores de intrusos.

A) Utilização de Sistemas Detectores de Intrusos:

Para combater os ataques a uma rede, sistemas de detecção de intrusão (SDI) são usados. Um SDI analisa a atividade de rede e tenta detectar qualquer comportamento incomum que possa afetar a integridade da rede. Com base nas informações fornecidas pelos SDI, as estratégias são criadas para lutar contra os ataques. Dentre os estudos primários selecionados 22% deles utilizam um SDI para garantir a segurança da Internet das Coisas.

Referência dos artigos contabilizados na subseção A:

- **Propõem ou usam SDI:** [Raza, Wallgren *and* Voigt, 2013; Vijay et al. 2015; Kang, Abbas *and* Oh, 2015; Sarigiannidis, Karapistoli *and* Economides, 2015; Desnitsky, Kotenko *and* Nogin, 2015; Christian et al. 2015; Christian et al. 2014].
- **Propõem ou usam SDI com Autonomia:** [Caiming et al. 2011].

B) Análise de protocolos de segurança e Modelos de segurança já existentes:

Após identificar um mal funcionamento da rede devido a um ataque é preciso analisar diversos fatores antes de tomar uma decisão. O ambiente da Internet das Coisas conta com protocolos específicos para redes com baixo recurso computacional. Então é importante que o consumo de energia e memória seja levado em consideração. Dentre os estudos primários selecionados 50% deles realizam análise dos protocolos de segurança, dos principais ataques no ambiente da Internet das coisas e de alguns modelos de segurança já existentes.

Referência dos artigos contabilizados na subseção B: [Granjal, Monteiro *and* Silva, 2015; Ngyuen, Laurent *and* Oualha, 2015; Ibrahim et al. 2015; Daniele et al. 2012; Bill, 2014; Zeng, Koutny *and* Watson, 2015; Arbia et al. 2014; Kumar, Vealey and Srivastava, 2016; Faiza et al. 2015; Benabdessalem, Hamdi, Kim, 2014; Hossain, Fotouhi *and* Hasan, 2015; Sadeghi,Wachsmann *and* Waidner, 2015; Xu, Wendt and Potkonjak, 2014; Convigton *and* Carskadden, 2013; Jiang *and* Shiwei, 2010; Aris, Oktug *and* Yalcin, 2015].

C) Modelos de segurança:

Tendo ciência dos recursos disponíveis e das possíveis ameaças, podem ser construídos alguns modelos de segurança que irão proporcionar confiança aos usuários das redes IoT. Dentre os estudos primários selecionados 28% deles propõem um modelo de segurança.

Referência dos artigos contabilizados na subseção C: [Jill et al. 2014; Thomas et al. 2013; Mohammad Sabzinejad et al. 2016; Turkanovic, Brumen *and* Hölbl, 2014; Yang, Forte *and* Tehranipoor, 2015; Johanna et al. 2016; Glowacka, Krygier *and* Amanowicz, 2015; Lei et al. 2015; Mališa et al. 2015].

4. CONCLUSÃO

Este trabalho teve como objetivo identificar e analisar as estratégias adotadas para tornar o ambiente da Internet das Coisas seguro. O processo de mapeamento sistemático foi conduzido por meio de um protocolo de busca e seleção de artigos que especificou a metodologia utilizada neste trabalho. Com os termos de busca definidos foram realizadas as buscas em inglês nas seguintes bases de dados: *IEEE Xplore (IEEE), Science Direct (SD);* e em português na base de dados: Biblioteca Digital Brasileira de Computação (BDBComp).

Ao final das buscas 258 artigos foram encontrados, 257 em inglês e 1 em português. Nos artigos encontrados foi realizada uma filtragem com uso dos critérios de seleção, os 34 artigos selecionados (33 em inglês e 1 em português) compuseram os estudos primários desse mapeamento sistemático. A partir da análise dos estudos primários, foi possível responder às questões de pesquisa levantadas, e assim, pode-se confirmar o aumento no ritmo de publicações na área de segurança da Internet das Coisas ao longo dos anos (Q1) e apresentar estratégias utilizados para tornar o ambiente em questão mais seguro (Q2).

Assim, acredita-se que esta pesquisa apresenta resultados relevantes à academia e aos empreendedores. Esse mapeamento pode ser estendido por meio da alteração de palavras-chave no termo de busca, alteração das questões de pesquisa ou dos critérios de inclusão e exclusão.

REFERÊNCIAS

GUBBI, Jayavardhana et al. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645-1660, 2013.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The internet of things: A survey. **Computer networks**, v. 54, n. 15, p. 2787-2805, 2010.

ROMAN, Rodrigo; NAJERA, Pablo; LOPEZ, Javier. Securing the internet of things. **Computer**, v. 44, n. 9, p. 51-58, 2011.

DOBSON, Simon et al. Fulfilling the vision of autonomic computing. **IEEE Computer**, 2010.

KEPHART, Jeffrey O.; CHESS, David M. The vision of autonomic computing. **Computer**, v. 36, n. 1, p. 41-50, 2003.

PETERSEN, Kai et al. Systematic mapping studies in software engineering. In: **12th international conference on evaluation and assessment in software engineering**. sn, 2008. p. 1-10.

RAZA, Shahid; WALLGREN, Linus; VOIGT, Thiemo. SVELTE: Real-time intrusion detection in the Internet of Things. **Ad hoc networks**, v. 11, n. 8, p. 2661-2674, 2013.

SIVARAMAN, Vijay et al. Network-level security and privacy control for smart-home IoT devices. In: **Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on**. IEEE, 2015. p. 163-167.

KANG, Chulhyun; ABBAS, Fizza; OH, Heekuck. Protection scheme for IoT devices using introspection. In: **Network of the Future (NOF), 2015 6th International Conference on the**. IEEE, 2015. p. 1-5.

LIU, Caiming et al. Research on immunity-based intrusion detection technology for the internet of things. In: **Natural Computation (ICNC), 2011 Seventh International Conference on**. IEEE, 2011. p. 212-216.

SARIGIANNIDIS, Panagiotis; KARAPISTOLI, Eirini; ECONOMIDES, Anastasios A. VisIoT: A threat visualisation tool for IoT systems security. In:**Communication Workshop (ICCW), 2015 IEEE International Conference on**. IEEE, 2015. p. 2633-2638.

DESNITSKY, V. A.; KOTENKO, I. V.; NOGIN, S. B. Detection of anomalies in data for monitoring of security components in the Internet of Things. In:**Soft Computing and Measurements (SCM), 2015 XVIII International Conference on**. IEEE, 2015. p. 189-192.

CERVANTES, Christian et al. Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things. In: **Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on**. IEEE, 2015. p. 606-611.

GRANJAL, Jorge; MONTEIRO, Edmundo; SILVA, Jorge Sá. Security in the integration of low-power wireless sensor networks with the internet: A survey.**Ad Hoc Networks**, v. 24, p. 264-287, 2015.

NGUYEN, Kim Thuat; LAURENT, Maryline; OUALHA, Nouha. Survey on secure communication protocols for the Internet of Things. **Ad Hoc Networks**, v. 32, p. 17-31, 2015.

MASHAL, Ibrahim et al. Choices for interaction with things on Internet and underlying issues. **Ad Hoc Networks**, v. 28, p. 68-90, 2015.

MIORANDI, Daniele et al. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, v. 10, n. 7, p. 1497-1516, 2012.

CURTIS, Bill. Delivering security by design in the Internet of Things. In: **Test Conference (ITC), 2014 IEEE International**. IEEE, 2014. p. 1-1.

ZENG, Wen; KOUTNY, Maciej; WATSON, Paul. Opacity in Internet of Things with Cloud Computing (Short Paper). In: **2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA)**. IEEE, 2015. p. 201-207.

RIAHI, Arbia et al. A systemic and cognitive approach for IoT security. In:**Computing, Networking and Communications (ICNC), 2014 International Conference on**. IEEE, 2014. p. 183-188.

KUMAR, Sathish Alampalayam; VEALEY, Tyler; SRIVASTAVA, Harshit. Security in Internet of Things: Challenges, Solutions and Future Directions. In: **2016 49th Hawaii International Conference on System Sciences (HICSS)**. IEEE, 2016. p. 5772-5781.

MEDJEK, Faiza et al. Analytical evaluation of the impacts of Sybil attacks against RPL under mobility. In: **Programming and Systems (ISPS), 2015 12th International Symposium on**. IEEE, 2015. p. 1-9.

BENABDESSALEM, Raja; HAMDI, Mohamed; KIM, Tai-Hoon. A Survey on Security Models, Techniques, and Tools for the Internet of Things. In:**Advanced Software Engineering and Its Applications (ASEA), 2014 7th International Conference on**. IEEE, 2014. p. 44-48.

HOSSAIN, Md Mahmud; FOTOUHI, Maziar; HASAN, Ragib. Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things. In: **Services (SERVICES), 2015 IEEE World Congress on**. IEEE, 2015. p. 21-28.

SADEGHI, Ahmad-Reza; WACHSMANN, Christian; WAIDNER, Michael. Security and privacy challenges in industrial internet of things. In:**Proceedings of the 52nd Annual Design Automation Conference**. ACM, 2015. p. 54.

XU, Teng; WENDT, James B.; POTKONJAK, Miodrag. Security of IoT systems: Design challenges and opportunities. In: **Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design**. IEEE Press, 2014. p. 417-423.

COVINGTON, Michael J.; CARSKADDEN, Rush. Threat implications of the internet of things. In: **Cyber Conflict (CyCon), 2013 5th International Conference on**. IEEE, 2013. p. 1-12.

JIANG, Du; SHIWEI, Chao. A study of information security for M2M of IOT. In: **Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on**. IEEE, 2010. p. V3-576-V3-579.

ARIS, Ahmet; OKTUG, Sema F.; YALCIN, Siddika Berna Ors. Internet-of-Things security: Denial of service attacks. In: **Signal Processing and Communications Applications Conference (SIU), 2015 23th**. IEEE, 2015. p. 903-906.

JERMYN, Jill et al. Firecycle: A scalable test bed for large-scale LTE security research. In: **Communications (ICC), 2014 IEEE International Conference on**. IEEE, 2014. p. 907-913.

KOTHMAYR, Thomas et al. DTLS based security and two-way authentication for the Internet of Things. **Ad Hoc Networks**, v. 11, n. 8, p. 2710-2723, 2013.

FARASH, Mohammad Sabzinejad et al. An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the Internet of Things environment. **Ad Hoc Networks**, v. 36, p. 152-176, 2016.

TURKANOVIĆ, Muhamed; BRUMEN, Boštjan; HÖLBL, Marko. A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion. **Ad Hoc Networks**, v. 20, p. 96-112, 2014.

YANG, Kun; FORTE, Domenic; TEHRANIPOOR, Mark M. Protecting endpoint devices in IoT supply chain. In: **Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on**. IEEE, 2015. p. 351-356.

SEP, Johanna et al. Dynamic NoC buffer allocation for MPSoC timing side channel attack protection. In: **2016 IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS)**. IEEE, 2016. p. 91-94.

GŁOWACKA, Joanna; KRYGIER, Jaroslaw; AMANOWICZ, Marek. A trust-based situation awareness system for military applications of the internet of things. In: **Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on**. IEEE, 2015. p. 490-495.

DONG, Lei et al. A Secure Collusion-Aware and Probability-Aware Range Query Processing in Tiered Sensor Networks. In: **Reliable Distributed Systems (SRDS), 2015 IEEE 34th Symposium on**. IEEE, 2015. p. 110-119.

VUČINIĆ, Mališa et al. OSCAR: Object security architecture for the Internet of Things. **Ad Hoc Networks**, v. 32, p. 3-16, 2015.

CERVANTES, Christian et al. Um Sistema de Detecç ao de Ataques Sinkhole sobre 6LoWPAN para Internet das Coisas. 2014.